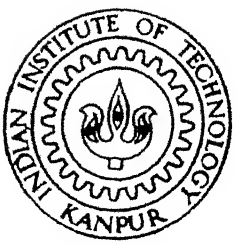
Construction of Elliptic Curves for Efficient and Secure Cryptosystems

by

PANKAJ BANSAL



DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

MARCH, 1997

Construction of Elliptic Curves for Efficient and Secure Cryptosystems

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Pankaj Bansal

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

INDIA

March 1997

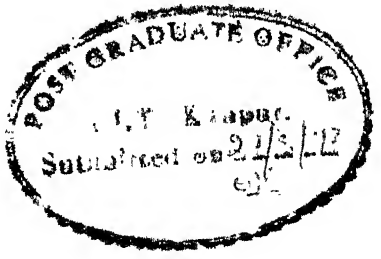
Abstract

Soon after Lenstra gave an algorithm for integer factorization using elliptic curves, Miller and Koblitz independently demonstrated the application of elliptic curves in public key cryptography. Initially the elliptic curve public key cryptosystems were thought to be impractical, but over the last 10 years, efforts of Menezes, Vanstone and Koblitz have changed the scenario. In contemporary cryptography, elliptic curve public key cryptosystems are being considered as a suitable alternative to existing RSA cryptosystems as they promise higher security with shorter keys. Shorter keys lead to reduction in storage requirements. Moreover, the complexity of hardware, required for implemention of elliptic curve public key cryptosystems, is relatively less as the size of working field is relatively small. Both of these factors make elliptic curves suitable for smart card implementation.

Security, throughput and complexity of the system are three important aspects of a cryptosystem design. In this thesis, we discuss the theory of elliptic curves to develop algorithms for constructing non-supersingular elliptic curves over finite fields which are suitable for secure cryptosystems. The algorithms are based on Lay & Zimmer's scheme (which is modified version of Atkin & Morain's scheme) and have been generalized to include several different cases. Relevant theory for construction of elliptic curves suitable for public key cryptosystems has been given in detail. The elliptic curves have also been discussed for certain number theoretic problems, such as integer factorization and primality proving. Thesis also discusses various issues related to efficient implementation of cryptosystems in general and smart cards in particular. Our major emphasis is on optimizing the memory requirements and complexity of modules, performing arithmetic in underlying finite field.

Certificate

This is to certify that the work contained in the thesis entitled **Construction of Elliptic Curves for Efficient and Secure Cryptosystems** by **Pankaj Bansal** has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



M. U. Siddiqi, 27 3 87
Professor,
Department of Electrical Engineering,
Indian Institute of Technology, Kanpur.

Contents

Chapter 1

Introduction

The fundamental goal of cryptography has been to provide secure communication over an insecure channel such that only intended recipient can read the message. Earlier, the scope of cryptography was considered to be limited to military and diplomatic communities, but in this age of universal electronic connectivity, the explosive growth in computer systems and their interconnections via networks has increased the dependence of both organizations and individuals on communicating information through an open medium. This together with increasing demand of automated financial transactions on networks has led to a heightened awareness of the need to protect data and resources from disclosure, which, in turn, has boosted the cause of cryptography. For some of the interesting applications of cryptography, see [Sta95, Sch93, Sim91, Omu90].

In cryptographic terminology, the process of encoding the data so as to hide its substance is called **encryption**, whereas the decoding process is called **decryption**. **Digital signatures**, electronic analog of handwritten signatures, are used for proving sender's identity to receiver. This process is called **authentication**. Authentication is often required whenever the access over any system is to be restricted.

Until 1976, when Diffie and Hellman [DH76, DH79] introduced the concept of public key cryptography, the symmetric key cryptosystems were the only means of achieving objectives of cryptography. In symmetric key cryptosystems, a single key is used for both encryption and decryption. Moreover, the sender and receiver must possess the secret key. This leads to the problem of key distribution as initially a secure channel is required to communicate the secret key. Moreover, the management of the keys is also a big problem as for any two users a different secret key is to be used. The public key cryptosystems solve both of these problems by using separate keys for encryption and decryption.

1.1 Public Key cryptography

The invention of public key cryptography by Diffie and Hellman added new dimension to the field of cryptography by introducing the use of two keys for encryption and decryption. The encryption key is made public, while the decryption key is kept secret. Since the encryption key is public, anybody can use it for encryption. However, only an authorized receiver who possesses the secret decryption key, is able to decrypt the message. Consequently, the public encryption key can be distributed without security concerns. This resolves the problem of distributing the keys. Key management also becomes easier because public keys are used for encryption which are not supposed to be kept secret. Public key cryptography also makes possible the use of digital signatures which is not possible in case of secret key systems.

Security of the public key cryptosystems depends upon the difficulty of finding the private key from public key. For this, Diffie and Hellman exploited intractability of the discrete logarithm problem in a large finite multiplicative group. This discrete logarithm problem (DLP) refers to computation of x such that $a^x = b$ for two given elements a and b in a finite group G. The value for x is called a logarithm of b to the base a, and is denoted by $\log_a b$. This problem is considered to be very difficult if the order of the group G is very large. In fact, the difficulty of this problem is also largely related to the representation of group and the complexity of binary operation defined in it.

Although, the Diffie and Hellman suggested only discrete logarithm problem for cryptography, this idea can be extended to any mathematical problem in which the forward computation is very easy but the reverse computation is known to be very difficult. Recognizing this very fact, in 1978, Rivest, Shamir and Adleman proposed RSA algorithm for public key cryptography which is based on the difficulty of factoring large composite numbers. In 1985, ELGamal gave an explicit methodology for using the DLP in implementing a fully functional cryptosystem [ElG85].

Other than cryptography, the idea of public key cryptography also had a sound impact on the research in computational number theory and other topics of mathematics. Nowadays, various problems of computational theory, i.e. integer factorization, primality testing, DLP etc, are being looked with great interest as the security of public key cryptosystems is directly related to them.

1.2 Implementation Issues of Cryptosystems

From the point of view of implementation, the symmetric key systems are preferred over public key cryptosystems as they are much faster and their VLSI implementation is very easy. Hence the use of hybrid system is also recommended in which encryption is done

with a symmetric key algorithm, but each time a different key is used and this key is sent along with the message by encrypting the secret key by a public key algorithm. But in applications which mainly require authentication it is desirable to have a cryptosystem on a single integrated chip which can perform the arithmetic required for public key algorithms.

Nowadays, the issue of major concern is to develop a smart card which is like a credit card but contains memory and processing capabilities. The smart card, in fact, is a multipurpose, tamper resistant security device which is equipped with volatile and non-volatile memory and a microprocessor for carrying out the computations for encryption and decryption. The basic advantage of the smart card is that the secret key of the user is stored in a nonvolatile memory and never leaves the card. All the processing required for encryption and decryption is done on the card itself. However, the kind of computations feasible by current cards are quite limited due to their relatively slow processing power and limited memory. Improving the speed of those devices is a very challenging task.

While selecting a public key algorithm for smart cards, various issues should be taken into consideration, most important ones being storage and processing requirements. Significant advances have also been made, in theory and in practice, on techniques for efficient implementations of these cryptosystems, including, custom VLSI chips [OSA99, VVDJ99, Bri89] and very efficient digital signal processor software implementations of modular arithmetic for RSA [DJ91], and custom VLSI chips for arithmetic operation in ElGamal cryptosystem in $GF(2^n)$ [AMOV91, WTS+85, WP90, Fen89]. Presently, the RSA is being used widely for smart cards as it requires modular exponentiation for both encryption and decryption. Keeping in mind the present state of computational technology, the RSA system requires modular arithmetic of at least 512 bit integers. So far various chips [OSA99, VVDJ99, Bri89] have been designed for this purpose and the maximum throughput achieved is in the range of 78kbps/20Mhz[1] [OSA99]. For decryption, higher throughputs have been claimed using Chinese remainder theorem which uses the (secret) factors of modulo integer for decryption [DJ91]. But under the recent improvement in integer factorization and parallel processing, the security of these systems is facing a serious threat. Recently, Lenstra et al were able to factor a 450 bit composite integer using distributed processing. Hence, for the cryptosystems to be secure the size of the modulo integers needs to be increased further. But this leads to various VLSI implementation related problems as the arithmetic with such large size integers is very difficult in hardware and moreover, more space will be required for key storage.

Because of all these reasons, alternatives are being looked with great interest. Since it is known that for a public key cryptosystem based on discrete logarithm problem in a multiplicative group the size of the group need to be approximately 2^{700}, they are no better

[1]encryption rate/clock frequency

than RSA system. Recent studies of discrete logarithm problems in elliptic curve groups suggest that elliptic curve cryptosystems will be a proper substitute for RSA in future as they give better security for shorter keys and are suitable for smart card implementations.

1.3 Elliptic Curve Cryptsystems

Elliptic curve cryptosystems were first proposed by V. Miller and N. Koblitz independently in 1985. The set of points on an elliptic curve over a finite field takes an abelian group structure. The binary operation for this abelian group involves few basic operations in the field over which the curve is defined. It was found by Miller and Koblitz that the discrete logarithm problem in an abelian group of elliptic curve over a finite field is much more complex than that over the field itself. Although there are several algorithms for finding logarithm in a finite field but there seems to be no proper choice for elliptic curve group (set of all the points) other than the few, which are applicable over any arbitrary group. Hence, an elliptic curve provides a group which is defined over a much smaller field. In other words it requires less storage and arithmetic of smaller size elements, and still gives better security than RSA. Drastic reduction in circuit complexity and storage requirements, and intractability of discrete logarithm problem in the elliptic curve group, make them a suitable candidate for smart card cryptosystems. Another advantage with elliptic curves is that for any given field there are plenty of choices for elliptic curves. Hence, a processor designed to perform the arithmetic in a finite field can be used in different elliptic curve based cryptosystems which requires arithmetic in that field.

1.4 Objective of Thesis

In this thesis, we discuss various design and implementation issues connected with elliptic curve cryptosystems. Our aim in this thesis is to construct elliptic curves which are secure against all the existing attacks and discuss various issues related to their efficient implementation. For this we discuss various techniques for efficient arithmetic in $GF(2^n)$ and $GF(p)$, both for software and hardware implementation. Reduction in the size of underlying field for elliptic curve cryptosystems makes them suitable for smart card implementations. We also discuss design aspects of elliptic curve based smart cards, specially the selection of finite fields and curves to minimize the storage and computational requirements. Apart from this, elliptic curves also provide efficient ways of factoring large integers and primality proving.

In this thesis, we pay considerable attention for the the theory of elliptic curves and their arithmetic to understand their applicability in both cryptography and cryptanalysis (integer factorization).

1.5 Organization of Thesis

The thesis is organized into 8 chapters, including the present one. Chapter 2 discusses the theory of elliptic functions and elliptic curves over $\mathbb{C}$ to obtain the elliptic curves over finite fields. In Chapter 3, we discuss the discrete logarithm problem in elliptic curve groups and its application to cryptography. Chapter 4 presents algorithms for construction of supersingular and non-supersingular elliptic curves for public key cryptography and also discusses various implementation issues. In Chapter 5, certain elliptic curve based number theoretic algorithms have been discussed. Chapter 6 deals with efficient implementation of elliptic curve public key cryptosystems and smart card design. Various implementation results have been discussed in Chapter 7. Chapter 8 concludes the thesis. The thesis also has four appendices A, B, C and D which give brief introduction to theory of modular forms, algebraic number theory, class field theory and algebraic geometry respectively.

Chapter 2

Theory of Elliptic Curves

The theory of elliptic curves is a beautiful and vast body of knowledge. The scope of elliptic curves stretches to various advance topics in mathematics like *Algebraic Number Theory, Algebraic Geometry, Complex Analysis* etc and hence, there are several ways to study them and their arithmetic. Our aim in this chapter is to study the theory of elliptic curves (over $\mathbb{C}$) to obtain elliptic curves over finite fields $GF(p)$ and $GF(2^n)$ for constructing efficient and secure cryptosystems. Here, we begin with the theory of elliptic functions in context to Weierstrass's work for solving the elliptic integral and then, concentrate over elliptic curves and their relationship with quadratic imaginary fields. In the first section, we discuss Weierstrass's elliptic function $\wp(z)$ as a tool for solving the elliptic integral problem.

2.1 Weierstrass's Elliptic Function $\wp(z)$

Geometrically, elliptic curve are not related to ellipse, as it appears from their name. But they evolve naturally in the process of computing the inverse of an elliptic integral [Seg80, Apo76, Cha88] which is come across in computing the arc length over an ellipse. The elliptic integral is an integral given by

$$\int \frac{dx}{\sqrt{x^3 + ax + b}} \qquad \text{where} \quad a, b \in \mathbb{C}$$

By computation of inverse of elliptic integral, we mean the computation of the angle subtended by an arc of given length over an ellipse. We will see that this problem helps us in setting up a relationship between elliptic curve over $\mathbb{C}$ and complex field $\mathbb{C}$ itself. Let us define the mermorphic function [Seg80, Ahl79] as the first step for solving this problem of inverting elliptic integral.

Definition 2.1 *A mermorphic function is an analytic function with isolated singularities over a region* Ω.

Our interest is in doubly periodic mermorphic functions [Apo76, Sil85, PS93, Seg80]

Definition 2.2 *The doubly periodic mermorphic functions are called elliptic functions.*

The set of all the periods of an elliptic function is called a lattice and is denoted by Λ. Any pair of periods (ω_1,ω_2) is called the basis of the lattice if any $\omega \in \Lambda$ has the unique representation $\omega = n_1\omega_1 + n_2\omega_2$ for $n_1, n_2 \in \mathbb{Z}$. Hence

$$\Lambda = \mathbb{Z}\omega_1 + \mathbb{Z}\omega_2$$

It is clear that the whole complex plane can be covered by a network of non-overlapping similar copies of (fundamental) parallelogram formed by vertices $0,\omega_1,\omega_2$ and $\omega_1+\omega_2$. The lattice Λ with basis (ω_1,ω_2) is, in fact, the set of vertices of this network of parallelograms [Seg80, Sil94].

Two lattices Λ_1, Λ_2 are called homothetic [Sil94] if $\Lambda_1 = c\Lambda_2$ for some $c \in \mathbb{C}$. If ω_1 and ω_2 are selected such that $\Im(\omega_1/\omega_2) > 0$ then Λ will be given by (upto homothety)

$$\Lambda_\tau = \mathbb{Z}\tau + \mathbb{Z}, \qquad \text{where} \quad \tau = \omega_1/\omega_2.$$

Since elliptic functions are periodic with ω_1 and ω_2, we will concentrate over the fundamental parallelogram only. The next lemma describes the uniqueness of a lattice [Apo76, Sil94].

Lemma 2.1 *(a) Let $\Lambda \in \mathbb{C}$ be a lattice and (ω_1,ω_2) and (ω_1',ω_2') be two oriented bases for Λ. Then*

$$\begin{aligned} \omega_1' &= a\omega_1 + b\omega_2 \\ \omega_2' &= c\omega_1 + d\omega_2, \end{aligned}$$

where $ad - bc = \pm 1$.

(b) If τ_1 and τ_2 lie in upper half of complex plane, then the lattices Λ_{τ_1} and Λ_{τ_2} are homothetic iff

$$\tau_2 = \frac{a\tau_1 + b}{c\tau_1 + d} \qquad \text{and} \quad ad - bc = \pm 1$$

(c) Every lattice Λ can be written upto homothety as $\Lambda_\tau = \mathbb{Z}\tau + \mathbb{Z}$ for some τ in upper half of complex plane.

Hence we see from this theorem that two lattices will be homothetic (similar in some sense) if and only if corresponding bases can be associated with a unimodular transformation. Now we state some of the important properties of of elliptic functions which will be required for the construction of Weierstrass elliptic function $\wp(z)$ [Ahl79, Seg80, Sil85].

properties of elliptic function with lattice Λ:

1. An elliptic function without poles is constant.
2. The sum of the residues of an elliptic function is zero.
3. A non constant elliptic function has equal number of poles and zeros, and zeros $a_1, a_2, \ldots, a_n$ and poles $b_1, b_2, \ldots, b_n$ satisfy

$$a_1 + a_2 + \ldots + a_n \equiv b_1 + b_2 + \ldots + b_n \mod \Lambda$$

4. The derivative of any elliptic function is also elliptic with the same periods, and hence, with the same lattice.
5. Elliptic functions takes all the values in $\mathbb{C}$.

In view of these properties, we construct an elliptic function with double poles at all the lattice points [Cha88, Seg80, Ahl79]. This function is called Weierstrass elliptic function and is denoted by $\wp(z)$.

$$\wp(z) = \frac{1}{z^2} + \sum_{\omega \in \Lambda, \omega \neq 0} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2}\right).$$

The Weierstrass function can easily be verified to possess all the properties mentioned above. Differentiating this formula, we get

$$\wp'(z) = -\sum_{\omega \in \Lambda} \frac{1}{(z-w)^3},$$

which is again an elliptic function with the same lattice, but with a third order pole at all the lattice points. By simple algebraic manipulation, it can easily be shown that following equation holds

$$(\wp'(z))^2 = 4\wp(z)^3 - g_2(\Lambda)\wp(z) - g_3(\Lambda),$$

where $g_2(\Lambda) = 60\mathrm{G}_4(\Lambda)$ and $g_3(\Lambda) = 140\mathrm{G}_6(\Lambda)$. We call this equation the elliptic curve equation [IR82, Cha88, Sil85, Ahl79]. The function $\mathrm{G}_{2k}(\Lambda)$ is a modular form of weight $2k$ and can be expressed in a series expansion (Eisenstien series)

$$\mathrm{G}_{2k}(\Lambda) = \sum_{\omega \in \Lambda, \omega \neq 0} \left(\frac{1}{\omega^{2k}}\right).$$

For a brief introduction to modular forms please see Appendix A.

Before proceeding further, we set an objective of studying the maps which will take an elliptic curve into itself. We shall discuss these maps in detail later on. Now, we come back to the problem of elliptic integral and see how it helps us in achieving this objective.

The Weierstrass elliptic curve equation is a first order differential equation. Substituting $x = \wp(z)$ and integrating both side, we get

$$z - z_0 = \int_{\wp(z_0)}^{\wp(z)} \frac{dx}{\sqrt{4x^3 - g_2(\Lambda)x - g_3(\Lambda)}},$$

where the path of integration is the image under $\wp$ of a path from z_0 to z that avoids the zeros and poles of $\wp'(z)$, and where the sign of the square root must be chosen so that it actually equals $\wp'(z)$. It should be noticed that integral on right hand side is an elliptic integral. Hence, we have solved the problem of inversion of elliptic integral because for an arc over an ellipse with end points z and z_0, the angle subtended at the center of ellipse will be given by $\wp(z) - \wp(z_0)$.

Since $\wp(z)$ takes all the values (property 5) in the complex plane (hence in fundamental parallelogram if taken modulo Λ), a one-to-one map exists between z and $\wp(z)$, both taking values in fundamental parallelogram when reduced modulo Λ. Here, it should be noticed that $\wp(z)$ along with $\wp'(z)$ (with an appropriate sign) defines a point on the elliptic curve E.

$$\text{E: } y^2 = 4x^3 - g_2(\Lambda)x - g_3(\Lambda),$$

where $x = \wp(z)$ and $y = \wp'(z)$.

The following theorem says that the one-to-one map mentioned above is, in fact, an isomorphism [Sil85].

Theorem 2.1 *(a) Let $\Lambda \in \mathbb{C}$ be the lattice generated by ω_1 and ω_2 (or equivalently 1 and τ, with $\tau = \omega_1/\omega_2$). Then the map*

$$\begin{aligned} \mathcal{F} : \mathcal{E}_\Lambda(\mathbb{C}) &\longrightarrow \mathbb{C}/\Lambda \\ \mathcal{P} &\longmapsto \mathcal{F}(\mathcal{P}) = \int_{\mathcal{O}}^{\mathcal{P}} \frac{dx}{y} mod\ \Lambda \end{aligned}$$

with $\mathcal{P} = (\wp(z), \wp'(z))$ and point at infinity $\mathcal{O}$, is a complex analytic isomorphism.

(b) It's inverse map is given by

$$\begin{aligned} \phi : \mathbb{C}/\Lambda &\longrightarrow E_\Lambda(\mathbb{C}) \\ z &\longmapsto \phi(z) = (\wp(z), \wp'(z)) \end{aligned}$$

This theorem is very important as it establishes an isomorphism between complex lie group $\mathbb{C}/\Lambda$ and E_Λ. Hence, now we will study complex lie group $\mathbb{C}/\Lambda$ to make deduction about the elliptic curve group E_Λ. Since the complex lie group of all the lattices, which are homothetic to Λ, is isomorphic to $\mathbb{C}/\Lambda$, the above theorem leads us to a very important result which says that elliptic curves corresponding to homothetic lattices will be isomorphic. The

generalization of this concept is quite immediate and can easily be extended to isomorphism classes of elliptic curves [Sil94]. If we denote by $\mathcal{L}$ the set of all the lattices in $\mathbb{C}$

$$\mathcal{L} = \{\text{Lattices in } \mathbb{C}\},$$

then the above discussion can be summarized by an isomorphism map

$$\mathcal{L}/\mathbb{C}^* \longrightarrow \frac{\{\text{elliptic curves defined over } \mathbb{C}\}}{\mathbb{C}\text{-isomorphism}} = \mathcal{E}\mathcal{L}\mathcal{L}_{\mathbb{C}}.$$

In the next section, we see how this map helps in achieving the objective we set earlier in this section.

2.2 Elliptic Curves over $\mathbb{C}$

According to discussion in section 2.1, if two lattices are homothetic then the corresponding elliptic curves will be isomorphic. Hence, if we want to determine whether two elliptic curves $E_1 : y^2 = x^3 + a_1x + b_1$ and $E_2 : y^2 = x^3 + a_2x + b_2$ are isomorphic or not, we need to first find the periods of corresponding lattices. This can be done by identifying basis (γ_1, γ_2) of homology group $H(\mathrm{E}(\mathbb{C}),\mathbb{Z})$ for E_1 and E_2 and using the map given in Theorem 2.1 for computation of basis (ω_1,ω_2) of corresponding lattices [Ahl79, Sil85].

$$\omega_1 = \int_{\gamma_1} \frac{dx}{y} \quad \omega_2 = \int_{\gamma_2} \frac{dx}{y}$$

If the corresponding lattices are homothetic then the two curves will be isomorphic.

Now, a natural question to be asked is whether there is simpler way or not. The answer to this question comes from the theory of modular forms [Apo76, PS93, Sil94, Seg80]. We know that *modular functions* (modular form of weight 0) are invariant functions for a homothety class of lattices (Please see Appendix A). Hence if we can obtain a modular function from modular forms $g_2(\Lambda)$ and $g_3(\Lambda)$, which are coefficients of elliptic curve equation, then we can obtain a parameter which will be invariant for an isomorphism class of elliptic curves. The Klein's $j(\Lambda)$, referred as j-invariant, is such a function.

$$j(\Lambda) = 1728\frac{g_2^3(\Lambda)}{\triangle(\Lambda)}$$

where $\triangle(\Lambda) = g_2^3(\Lambda) - 27g_3^2(\Lambda)$ is called discriminant and is a modular form of weight 12.

Now, let's see *Uniformization Theorem* which, together with the above discussion, gives the desired result [Sil85, Sil94].

Theorem 2.2 (Uniformization theorem of elliptic curves over $\mathbb{C}$) *Let $a, b \in \mathbb{C}$ satisfy $4a^3 + 27b^2 \neq 0$, then there is a unique lattice $\Lambda \in \mathbb{C}$ such that*

$$g_2(\Lambda) = -4a \quad \text{and} \quad g_3(\Lambda) = -4b.$$

The map

$$\begin{aligned} \mathbb{C}/\Lambda &\longrightarrow E: y^2 = x^3 + ax + b, \\ z &\longrightarrow (\wp(z), \wp'(z)/2) \end{aligned}$$

is a complex analytic map.

This theorem does the important task of parameterizing the curve. The discriminant and j-invariant of the curve E are given by

$$\triangle(E) = -16(4a^3 + 27b^2) \quad \text{and } j(E) = -1728(4a)^3/\triangle(E)$$

Now, since any elliptic curve over $\mathbb{C}$ is uniquely identified by its j-invariant, E_1 and E_2 will be isomorphic if and only if their j-invariants are same. The next proposition summarizes much of the preceding discussion [Sil94].

Proposition 2.1 *There are one-to-one correspondences between the following sets, given by identical maps*

$$\begin{aligned} \mathcal{ELL}_{\mathbb{C}} &\longleftarrow \mathcal{L}/\mathbb{C}^* \longrightarrow \mathbb{C} \\ \{E_\Lambda\} &\longleftarrow \{\Lambda = \Lambda_\tau\} \longrightarrow j(\tau) \end{aligned}$$

Here $\Lambda_\tau = \mathbb{Z}\tau + \mathbb{Z}$, $\{E_\Lambda\}$ denotes the $\mathbb{C}$-isomorphism classes of elliptic curves $E_\Lambda : y^2 = 4x^3 - g_2(\Lambda)x - g_3(\Lambda)$, and $\{\Lambda\}$ is the homothety class of Λ.

Before discussing another very important property of j-invariant, let us first define *field of modulii* [Sil94].

Definition 2.3 *Let $\{E\} \in \mathcal{ELL}_{\mathbb{C}}$ and $\mathcal{K} \in \mathbb{C}$. We say that $\mathcal{K}$ is the field of definition for $\{E\}$ if there is an elliptic curve $E_0 \in \{E\}$ such that E_0 is defined over $\mathcal{K}$. We say that $\mathcal{K}$ is the* field of modulii *for $\{E\}$ if for all the automorphisms $\sigma \in Aut(\mathbb{C}/\mathbb{Q})$*

$$E^\sigma \in \{E\} \qquad \text{iff } \sigma \text{ acts trivially on } \mathcal{K}$$

Though this definition does not guarantee the existence and uniqueness of field of modulii for an isomorphism class of elliptic curves, it is so. In fact, there are finite number of isomorphism classes which have the same field as field of modulii and all of these isomorphism classes have complex multiplication by an order[1] in imaginary quadratic field (explained later). The next

[1]The term order is used in two contexts in this thesis, for a ring in imaginary quadratic field and for number of point of elliptic curves.

proposition, the proof of which is quite elementary, gives a very important result which relates the j-invariant of the isomorphism class to the field of modulii.

Proposition 2.2 *Let* $\{E\} \in \mathcal{ECC}_{\mathbb{C}}$.

(a) $\mathbb{Q}(j\{E\})$ *is the field of modulii for* $\{E\}$.

(b) $\mathbb{Q}(j\{E\})$ *is the minimal field of definition for* $\{E\}$.

The importance of this proposition will become obvious later when we will identify an isomorphism class of elliptic curves by an elliptic curve over its field of modulii $\mathbb{Q}(j\{E\})$. Now, we come to the objective set in Section 2.1, i.e. study of maps on elliptic curves [TV91, Sil85, Mor93].

Definition 2.4 *Let* E_1 *and* E_2 *be elliptic curves. An isogeny between* E_1 *and* E_2 *is a morphism*

$$\phi : E_1 \longrightarrow E_2$$

satisfying $\phi(\mathcal{O}) \longmapsto \mathcal{O}$. *Here* $\mathcal{O}$ *is point at infinity.*

We shall see later that E takes a structure of group and $\mathcal{O}$ acts as identity in that group. Presently, we proceed further with this assumption.
The set of all the isogenies, which take an elliptic curve into itself, i.e. set of all the endomorphisms, takes a ring structure in a natural way and is called endomorphism ring of E [Ono90, Sil85, Mor93, Cha88].

$$End(E) = \{\phi : \phi(E) \subseteq E\}$$

As stated earlier, we are interested in studying the endomorphism ring of an elliptic curve. Now the importance of Theorem 2.1 is clear as the isomorphism between E/Λ and $\mathbb{C}/\Lambda$ shows that we can, instead, study all the homomorphisms of $\mathbb{C}/\Lambda$ and can make deduction about the endomorphism ring of E/Λ. This will lead us to a quadratic imaginary field through the notion of *complex multiplication* [PS92, Sil85, Sil94, BCh$^+$66, Shi71].

Let Λ_1 and Λ_2 be lattices in $\mathbb{C}$. If $\alpha \in \mathbb{C}$ has the property that $\alpha\Lambda_1 \subset \Lambda_2$, then the scalar multiplication by α

$$\begin{aligned} \phi_\alpha : \mathbb{C}/\Lambda_1 &\longrightarrow \mathbb{C}/\Lambda_2 \\ z &\longmapsto \phi_\alpha(z) = \alpha z \bmod \Lambda_2 \end{aligned}$$

is clearly an analytical homomorphism. The next theorem shows that these are essentially the only holomorphic (analytic) maps [PS92, Sil85, Cha88].

Theorem 2.3 *(a) The association*

$$\begin{aligned} \{\alpha \in \mathbb{C} : \alpha\Lambda_1 \subset \Lambda_2\} &\longrightarrow \{\textit{holomorphic maps } \phi : \mathbb{C}/\Lambda_1 \to \mathbb{C}/\Lambda_2 \quad \phi(\mathcal{O}) = \mathcal{O}\} \\ \alpha &\longrightarrow \phi_\alpha \end{aligned}$$

is a bijection.

(b) Let E_1 and E_2 be the elliptic curves corresponding to two lattices Λ_1 and Λ_2. Then the natural inclusion

$$\{\textit{isogenies } \alpha : E_1 \to E_2\} \longrightarrow \{\textit{holomorphic maps } \phi : \mathbb{C}/\Lambda_1 \to \mathbb{C}/\Lambda_2 \quad \phi(\mathcal{O}) = \mathcal{O}\}$$

is a bijection.

This theorem strengthens the foregoing argument and motivates us to study the set of all endomorphisms of complex lie groups $\mathbb{C}/\Lambda$ to determine the structure of endomorphism ring of E.

2.3 Complex Multiplication

Now, we shall study elliptic curves with complex multiplication. This section requires elementary knowledge of algebraic number fields (esp. Quadratic imaginary fields) [PS93, Ros94, IR82, Hec93] and their abelian extensions, i.e. class fields [PS92, Coh78, BCh$^+$66, Sil94]. An introductory overview of these topics is given in Appendices B and C. We begin with a theorem which will help in defining the term *complex multiplicatio* for elliptic curves [Sil85, BCh$^+$66, PS92].

Theorem 2.4 *(a) Let E be an elliptic curve over a field K and let $m \in \mathbb{Z}$, $m \neq 0$. Then the multiplication by m map*

$$[m] : E \longrightarrow E$$

is non-constant.

(b) The endomorphism ring $End(E)$ is an integral domain of characteristic 0.

(c) The endomorphism ring of an elliptic curve is either isomorphic to $\mathbb{Z}$ or an order in a quadratic imaginary field or an order in quaternion algebra.

It can be shown for $K = \mathbb{C}$ that $End(E)$ can not be an order in quaternion algebra. If K is a finite field then $End(E)$ will be strictly larger than $\mathbb{Z}$ [Shi71, Cha88, Sil85, PS93].

Definition 2.5 *If for any elliptic curve the endomorphism ring is larger than $\mathbb{Z}$, then we say that the elliptic curve has complex multiplication or CM in short.*

Elliptic curves, with their endomorphism ring isomorphic to an order in quaternion algebra, are called supersingular elliptic curves [Men93b, Sil85]. Since our final goal is to construct non-supersingular elliptic curves over finite field (Chapter 3 and 4), we concentrate over elliptic curves with complex multiplication.

To explain the isomorphism between $End(E)$ and an order $\mathcal{O}$ in an imaginary quadratic field $\mathcal{K}$, we come back to what we discussed at the end of Section 2.2. If we can somehow find all $\alpha \in \mathbb{C}^*$ such that

$$\phi_\alpha : \mathbb{C}/\Lambda \longrightarrow \mathbb{C}/\Lambda,$$

then that set (in fact, ring) will be isomorphic to $End(E)$. In other words, we are interested in finding all $\alpha \in \mathbb{C}^*$ such that

$$\alpha\Lambda \subseteq \Lambda.$$

Here, we need to introduce the quadratic imaginary field (See Appendix B) [Ono90, Ros94, Dav80]. It is not difficult to observe that each fractional ideal in an order $\mathcal{O}$ in any quadratic imaginary field $\mathcal{K}$, is equivalent to a lattice [Sil85, Cha88, Hec93, Ros94]. The basis of any $\mathcal{O}$-fractional ideal will be same as that of a lattice. Now if, instead of lattice, we consider an $\mathcal{O}$-fractional ideal of an appropriate imaginary quadratic field $\mathcal{K}$ then by the definition of the $\mathcal{O}$-fractional ideals, the required set will obviously be order $\mathcal{O}$. Hence, we have proved following corollary [Sil94].

corollary 2.1 *Let $E/\mathbb{C}$ be an elliptic curve with complex multiplication by a ring $\mathcal{O} \in \mathbb{C}$. Then the following map is an isomorphism*

$$[\cdot] : \mathcal{O} \longrightarrow End(E)$$

and the ring $\mathcal{O}$ can be identified as an order in an quadratic imaginary field $\mathcal{K}$.

Now, we will concentrate on isomorphism classes of elliptic curves which have same endomorphism ring, i.e. have complex multiplication by same order $\mathcal{O}$ [Sil94].

$$\begin{aligned} \mathcal{ELL}(\mathcal{O}) &= \frac{\{\text{Elliptic curves E}/\mathbb{C} \text{ with } End(\text{E}) \cong \mathcal{O}\}}{\text{isomorphisms over } \mathbb{C}} \\ &= \frac{\{\text{Lattices } \Lambda \text{with } \{\alpha : \alpha\Lambda \subseteq \Lambda,\ \alpha \in \mathbb{C}\} = \mathcal{O}\}}{homothety} \end{aligned}$$

Since each $\mathcal{O}$-fractional ideal can be looked at as a lattice, each non zero $\mathcal{O}$-fractional ideal can be associated with an elliptic curve with complex multiplication by $\mathcal{O}$. On the other hand, since homothety class of lattices gives a set of isomorphic elliptic curves, $\mathcal{O}$-fractional ideals $\mathfrak{a}$ and $c\mathfrak{a}$ ($c \in \mathcal{O}$) will give the same elliptic curve in $\mathcal{ELL}(\mathcal{O})$. Hence we can easily deduce that each ideal class in class group of order $\mathcal{O}$ can be associated with an isomorphism class of elliptic curves in a unique way [PS93, Sil85, Shi71, BCh+66].

If $\mathfrak{a}$ is a fractional ideal of a quadratic imaginary field $\mathcal{K}$, we denote by $\bar{\mathfrak{a}}$ its ideal class in $\mathcal{CL}(\mathcal{O})$. The following map is a bijection

$$\begin{aligned} \mathcal{CL}(\mathcal{O}) &\longrightarrow \mathcal{ELL}(\mathcal{O}) \\ \bar{\mathfrak{a}} &\longrightarrow E_{\mathfrak{a}} \end{aligned}$$

Now, we are heading towards the main result which is related to ring class field. In the literature usually, the main result is stated first and then proved. But, since the proof of main result is very complicated, we shall follow a rather different approach such that the statement of main result evolves naturally and does not require any proof. This surely requires good understanding of the concept of algebraic number fields and class fields.

We know that we can associate a unique class field with every order in $\mathcal{K}$ which is called *ring class field* [Coh78, PS93]. If the order is maximal, i.e. ring of integers $\mathcal{O}_{\mathcal{K}}$, then the ring class field will be Hilbert class field, a maximal unramified abelian extension of $\mathcal{K}$. In case of non maximal order, the class field will be be ramified abelian extension of $\mathcal{K}$ such that any $\mathcal{O}_{\mathcal{K}}$-integral may ramify in the concerned ring class field but all $\mathcal{O}$-integral will be unramified.

Let the ring class field be given by $\mathcal{H}_{\mathcal{O}}$. Then the following map will be an isomorphism (Please see Appendix C) [Ono90, Coh78, PS93, PS92].

$$\mathcal{F} : \mathrm{Gal}(\mathcal{H}_{\mathcal{O}}/\mathcal{K}) \longrightarrow \mathcal{CL}(\mathcal{O})$$

Hence, the action of $\mathrm{Gal}(\mathcal{H}_{\mathcal{O}}/\mathcal{K})$ on $\mathcal{ELL}(\mathcal{O})$ can be identified by action of $\mathcal{CL}(\mathcal{O})$ on $\mathcal{ELL}(\mathcal{O})$. The next proposition shows that the action of $\mathcal{CL}(\mathcal{O})$ on $\mathcal{ELL}(\mathcal{O})$ is simply transitive [Sil94, Shi71].

Proposition 2.3 *(a) Let Λ be a lattice with $E_{\Lambda} \in \mathcal{ELL}(\mathcal{O})$, and $\mathfrak{a}$ and $\mathfrak{b}$ be two non zero fractional ideals of $\mathcal{K}$. Then*

1. *$\mathfrak{a}\Lambda$ is a lattice in $\mathbb{C}$,*
2. *the elliptic curve $E_{\mathfrak{a}\Lambda}$ satisfies $End(E_{\mathfrak{a}\Lambda}) \cong \mathcal{O}$,*
3. *$E_{\mathfrak{a}\Lambda} \cong E_{\mathfrak{b}\Lambda}$ iff $\bar{\mathfrak{a}} = \bar{\mathfrak{b}}$ in $\mathcal{CL}(\mathcal{O})$.*

Hence, this is a well defined action of $\mathcal{CL}(\mathcal{O})$ on $\mathcal{ELL}(\mathcal{O})$ determined by

$$\bar{\mathfrak{a}} * E_{\Lambda} = E_{\mathfrak{a}^{-1}\Lambda}$$

(b) The action of $\mathcal{CL}(\mathcal{O})$ on $\mathcal{ELL}(\mathcal{O})$, described in (a) is simply transitive. In particular

$$\#\mathcal{CL}(\mathcal{O}) = \#\mathcal{ELL}(\mathcal{O})$$

In view of this proposition, we can see that action of $\mathrm{Gal}(\mathcal{H}_\mathcal{O}/\mathcal{K})$ on $\mathcal{ELL}(\mathcal{O})$, if looked as an action of $\mathcal{CL}(\mathcal{O})$ on $\mathcal{ELL}(\mathcal{O})$, would take E_Λ to $E_{\bar{\mathfrak{a}}^{-1}\Lambda}$ for fractional ideal $\bar{\mathfrak{a}}$ from the ideal class corresponding to concerned element of $\mathrm{Gal}(\mathcal{H}_\mathcal{O}/\mathcal{K})$. Hence, this action takes an elliptic isomorphism class to its conjugate isomorphism class. Since action of $\mathrm{Gal}(\mathcal{H}_\mathcal{O}/\mathcal{K})$ on E_Λ can also be looked as action on coefficients of elliptic curve equation, we can deduce that coefficients of elliptic curve equation, which are modular forms of weight 4 and 6, should lie in ring class field $\mathcal{H}_\mathcal{O}$. This deduction arises from the fact that action of $\mathrm{Gal}(\mathcal{H}_\mathcal{O}/\mathcal{K})$ on any element of $\mathcal{H}_\mathcal{O}$ will take that element to its conjugate [Ono90, Coh78, Fra95, PS92]. We present the above discussion in the next proposition.

Proposition 2.4 *Let $\mathcal{K}/\mathbb{Q}$ be a quadratic imaginary field. Then there exists an isomorphism*

$$\mathcal{F} : Gal(\mathcal{H}_\mathcal{O}/\mathcal{K}) \longrightarrow \mathcal{CL}(\mathcal{O})$$

uniquely characterized by the condition

$$E^\sigma = \mathcal{F}(\sigma) * E \quad \text{for all } \sigma \in Gal(\mathcal{H}_\mathcal{O}/\mathcal{K}) \text{ and all } E \in \mathcal{ELL}(\mathcal{O})$$

or equivalently $\quad j(E)^\sigma = j(\mathcal{F}(\sigma) * E).$

Recalling, the definition of *field of definition*, we can say that for all elliptic curves with complex multiplication by $\mathcal{O}$, the ring class field (not the field of modulii $\mathbb{Q}(j(\mathrm{E}))$ will be the field of definition. The next theorem, which was earlier referred as main result, tells us about the association between j-invariants and $\mathcal{H}_\mathcal{O}$ [Shi71, BCh+66, Sil94].

Theorem 2.5 *Let E be an elliptic curve representing an isomorphism class in $\mathcal{ELL}(\mathcal{O})$.*

(a) $j(E)$ is an algebraic integer and $\mathcal{K}(j(E))$ is the ring class field $\mathcal{H}_\mathcal{O}$ of $\mathcal{K}$.

(b) $[\mathbb{Q}(j(E)):\mathbb{Q}] = [\mathcal{K}(j(E)):\mathcal{K}] = h_\mathcal{K}$ where $h_\mathcal{K} = \#\mathcal{CL}(\mathcal{O}) = \#Gal(\mathcal{H}_\mathcal{O}/\mathcal{K})$

(c) Let $E_1 \ldots E_{h_\mathcal{K}}$ be a complete set of representatives for $\mathcal{ELL}(\mathcal{O})$, then $j(E_1) \ldots j(E_{h_\mathcal{K}})$ is a complete set of $Gal(\mathcal{H}_\mathcal{O}/\mathcal{K})$ conjugates of $j(E)$.

From this theorem, we can deduce that field of modulii $\mathbb{Q}(j(\mathrm{E}))$ for all the elliptic curves, having complex multiplication by $\mathcal{O}$, is a subfield of $\mathcal{K}(j(\mathrm{E}))$ and $[\mathcal{K}(j(\mathrm{E})):\mathbb{Q}(\ \mathrm{j}\ (E))] = 2$. The next section explains the reduction map for obtaining elliptic curves over finite field $GF(q)$.

2.4 Elliptic Curve Reduction Maps

In Section 2.3, we discussed the relationship between imaginary quadratic field and elliptic curves with the notion of complex multiplication and ring class field. In this section, we see

how an elliptic curve (over $\mathbb{C}$) can be reduced to an elliptic over finite field and how the reduction map will affect the order (number of points on curve) of the elliptic curve [Sil94, LZ94].

If any elliptic curve is nonsingular then it is said to have good reduction [BCh+66, PS93, Sil85]. Now, since any elliptic curve complex multiplication is defined over ring class field of an order in a quadratic imaginary field $\mathcal{K}$, we will see how to obtain a finite field from ring class field [Ono90].

Since, the degree of extension of $\mathcal{K}$ over $\mathbb{Q}$, $[\mathcal{K}:\mathbb{Q}]$, is 2, any prime p which splits completely in $\mathcal{K}$ will have degree 1. Let's say it splits in prime ideals $\mathfrak{p}$ and $\mathfrak{p}'$ in $\mathcal{K}$. Then

$$(p)_{\mathcal{K}} = \mathfrak{p}\mathfrak{p}' \quad \text{where} \quad \mathfrak{p} = \mathrm{N}_{\mathbb{Q}}^{\mathcal{K}}(\mathfrak{p}).$$

Now, if ring class field associated with an order $\mathcal{O}$ in $\mathcal{K}$ is given by $\mathcal{H}_{\mathcal{O}}$ and $\mathcal{O}_{\mathcal{H}_{\mathcal{O}}}$ denotes the maximal order in $\mathcal{H}_{\mathcal{O}}$, then $\mathfrak{p}$ will split in $\mathcal{O}_{\mathcal{H}_{\mathcal{O}}}$ into prime ideals $\mathfrak{P}_1, \ldots, \mathfrak{P}_g$ as

$$(\mathfrak{p})_{\mathcal{H}_{\mathcal{O}}} = \mathfrak{P}_1, \mathfrak{P}_2, \ldots, \mathfrak{P}_g.$$

Here the degree of $\mathfrak{P}_i$ over $\mathcal{K}$ is f [Hec93, IR82, PS92, Ono90, Cha88]. If the class number of of imaginary quadratic field $\mathcal{K}$ is given by $h_{\mathcal{K}}$ then,

$$h_{\mathcal{K}} = [\mathcal{H}_{\mathcal{O}} : \mathcal{K}] = fg.$$

Obviously, the degree of $\mathfrak{P}_i$ over $\mathbb{Q}$ will be

$$f_{\mathfrak{P}_i/\mathfrak{p}} = degree\ of\ \mathfrak{p} * f.$$

The residue operation $\mathcal{O}_{\mathcal{H}_{\mathcal{O}}}/\mathfrak{P}_i$ will give a field (Appendix C) of order

$$\mathrm{N}_{\mathbb{Q}}^{\mathcal{H}_{\mathcal{O}}}(\mathfrak{P}_i) = \mathrm{N}_{\mathbb{Q}}^{\mathcal{K}}(\mathfrak{p}^f) = p^f,$$

and characteristic p.

So now, we have obtained the finite field by residue action of $\mathfrak{P}_i$ on $\mathcal{O}_{\mathcal{H}_{\mathcal{O}}}$. As our final objective is to obtain elliptic curves over finite fields, the effect of this quotient operation on the elliptic curve, having that ring class field as field of definition, will surely be of interest to us. Since the coefficients of elliptic curve equation are from the ring class field $\mathcal{H}_{\mathcal{O}}$, reduction operation will obviously give a curve over finite field obtained as discussed above.

The next theorem is very important as it tells us about the effect of reduction map on elliptic curves [LZ94, Sil85].

Theorem 2.6 *(a) Let $E/\mathbb{C}$ be an elliptic curve with complex multiplication. Its j-invariant is an algebraic integer.*

(b) $E/\mathbb{C}$ has good reduction iff its j-invariant is an algebraic integer.

(c) The endomorphism ring of $E/\mathbb{C}$ is stable under the reduction map $E \to \tilde{E}$ by $\mathfrak{P}_i$ (as given above)i.e. $End(E) = End(\tilde{E})$.

For the integarality of j, [Sil94] contains various proofs. In fact, $\mathbb{Q}(j(\mathrm{E}))$ is the splitting field of $j(\mathrm{E})$, i.e. the polynomial

$$\prod_{\sigma \in Gal(\mathbb{Q}(j(E))/\mathbb{Q})} (x - j(E^{\sigma}))$$

has coefficients in $\mathbb{Z}$, and completely splits in $\mathbb{Q}(j(\mathrm{E}))$. A careful study reveals that $\mathcal{K}(j(\mathrm{E}))$ will be splitting field of this polynomial over $\mathcal{K}$. Hence we can see that Theorem 2.6 allows us to apply the reduction map [Ono90].

So, now we have obtained an elliptic curve over finite field $GF(q)$ of $q = p^f$ elements. The next important thing to be discussed is the order of the elliptic curve defined over finite field. Obviously the order, written as $\#\mathrm{E}(GF(q))$, will be finite as the curve is defined over a finite field $GF(q)$. Since the order of elliptic curve over $\mathbb{C}$ was infinite, we will now see how the reduction map defined above helps in determining order of elliptic curve over finite field. For this we need to introduce the Frobenius map [Ful69, Sil85] for algebraic curves, which drags us into algebraic geometry [Ful69, Sil85, Abh90, Wal62, Mor93].

Let $C/GF(q)$ a smooth (non singular) curve , then the map ϕ is called q^{th} power Frobenius morphism if

$$\begin{aligned} \phi : C &\longrightarrow C^{(q)} \\ \phi : [x, y] &\longrightarrow [x^q, y^q]. \end{aligned}$$

This map is purely separable and $deg(\phi) = q$ [Abh90, Ful69, Sil85].

Proposition 2.5 *Let $\phi \in End(E)$. Then*

$$Tr(\phi_\ell) = 1 + deg(\phi) - deg(1 - \phi)$$

Here, ϕ_ℓ is 2×2 matrix, a map for Tate module over *ℓ-adic* [Sil85, PS93] integers and Tr is a trace function over this matrix. By the abuse of notation, we write the above equation as

$$Tr(\phi) = 1 + deg(\phi) - deg(1 - \phi),$$

because it simplifies the equation and avoids the need of studying the Tate module theory which, again, in itself is very vast [Shi71, Sil94].

Now, we can easily see that for a point $\mathcal{P}(x, y) \in E/GF(q)$

$$\phi(\mathcal{P}) = (x^q, y^q) = (x, y) = \mathcal{P}$$

because in field $GF(q)$ $x^q = x$. Hence

$$\begin{aligned} & (1-\phi)(\mathcal{P}) = \mathcal{O} && (\mathcal{O} \text{ is point at infinity}) \\ \Rightarrow \quad & \mathrm{E}/GF(q) = Ker(1-\phi) \\ \textit{and} \quad & \#\mathrm{E}(GF(q)) = \#Ker(1-\phi) && (2.1) \\ \textit{similarly} \quad & \#\mathrm{E}(GF(q^n)) = \#Ker(1-\phi^n). && (2.2) \end{aligned}$$

Here,$(1-\phi)$ is a purely separable map [Sil85] and for a purely separable map $\psi(=1-\phi)$

$$\begin{aligned} & \#Ker(\psi) = deg(\psi) \\ \Rightarrow \quad & \#Ker(1-\phi) = deg(1-\phi) \end{aligned}$$

Combining this with Equation 2.1, we conclude

$$\begin{aligned} \#\mathrm{E}(GF(q)) &= \#Ker(1-\phi) \\ &= deg(1-\phi) \\ &= 1 + deg(\phi) - Tr(\phi). \end{aligned}$$

The next theorem gives us the final result [LZ94, Sil94].

Theorem 2.7 *Let $GF(q)$ be a quadratic imaginary field, $\mathcal{H}_\mathcal{O}$ the ring class field of $\mathcal{K}$, and $E/\mathcal{H}_\mathcal{O}$ an elliptic curve with complex multiplication by an order $\mathcal{O}$ in $\mathcal{K}$. Let $\sigma_\mathfrak{p} \in Gal(\mathcal{H}_\mathcal{O}/\mathcal{K})$ be the Frobenius element associated to a prime ideal $\mathfrak{p}$ of $\mathcal{O}$, and let $\mathfrak{P}_i$ be a prime ideals of $\mathcal{H}_\mathcal{O}$ lying above $\mathfrak{p}$. If $\mathfrak{p}$ has degree 1 and E has good reduction, then there exist an isogeny*

$$\phi : E \longrightarrow E^{\sigma_\mathfrak{p}}$$

whose reduction modulo $\mathfrak{P}_i$

$$\tilde{\phi} : \tilde{E} \longrightarrow \tilde{E}^{(q)}$$

is the q^{th} power Frobenius map. Moreover, there exists a unique $\pi = \pi_\mathfrak{p} \in \mathcal{O}$ such that

$$\mathfrak{p}^f = \pi\mathcal{O} \quad \textit{and} \quad \begin{array}{ccc} E & \xrightarrow{[\pi]} & E \\ \downarrow & & \downarrow \\ \tilde{E} & \xrightarrow[\substack{Frobenius\ map}]{q^{th}\ power} & \tilde{E} \end{array}$$

is commutative. Here, f is the degree of $\mathfrak{P}_i$ above $\mathfrak{p}$.

Here, an important thing to notice is that we have identified an element in $\mathcal{O}$ corresponding to Frobenius map associated with $\mathfrak{P}_i$ in $\mathcal{H}_\mathcal{O}$. In fact, the Frobenius map for $\mathfrak{P}_i$ is $\mathrm{N}_\mathbb{Q}^\mathcal{K}(\mathfrak{p})$ and π is f^{th} power of the norm of $\mathfrak{P}_i$ relative to $\mathcal{K}$ [Ono90, Hec93, PS92, IR82]. In other

words, $\mathfrak{p}^f$ is a principal ideal and π is an uniformizer element such that $\mathfrak{p}^f = \pi\mathcal{O}$. This leads us to a very important observation about the degree, f, of $\mathfrak{P}_i$ over $\mathfrak{p}$. The degree f is the minimal integer such that $\mathfrak{p}^f$ becomes principal ideal. We will see the implications of this observation later when we discuss the implementation aspects of the theory, discussed in this chapter, for constructing elliptic curves following certain constraints for security.

Combining the previous results, we can say that there exists a unique element $\pi = \pi_q \in \mathcal{O}$ (upto multiplication by an unit in $\mathcal{O}$) such that

$$q = deg(\phi) = p^f = \mathrm{N}_{\mathbb{Q}}^{K}(\pi)$$

and

$$\begin{aligned} \#\tilde{\mathrm{E}}(GF(q)) &= 1 + deg(\phi) - Tr(\phi) \\ &= 1 + \mathrm{N}_{\mathbb{Q}}^{K}(\pi) - Tr(\pi) \\ &= \mathrm{N}_{\mathbb{Q}}^{K}(1 - \pi) \end{aligned}$$

From these two equations (referred as norm equations), we can compute the order of the elliptic curve $\#\mathrm{E}(GF(q))$ and the finite field $GF(q)$ over which the reduced curve is defined. It should be noted that for a given q, i.e. the field, there will be more than one value for order of elliptic curves having complex multiplication with same order in an imaginary quadratic field and they will be equal to number of units in order $\mathcal{O}$. Hence this leads us to a very important result about the isomorphism of elliptic curves over finite fields.

Theorem 2.8 *Two elliptic curves $E_1(GF(q))$ and $E_2(GF(q))$ are isomorphic if and only if their order and j-invariant both are same.*

Obviously, the modulo $\mathfrak{P}_i$ map will reduce the Weierstrass elliptic curve equation

$$\mathrm{E}(\mathbb{C}) : y^2 = x^3 + ax + b, \qquad where \quad a, b \in \mathbb{C}$$

to

$$\mathrm{E}(GF(q)) : y^2 = x^3 + \hat{a}x + \hat{b} \qquad where \quad \hat{a}, \hat{b} \in GF(q)$$

The two norm equations, obtained above, will prove to be very important in constructing elliptic curves with predefined order because if we specify the finite field $GF(q)$ and an integer for order of the curve to be constructed then the solution of these equations in an appropriate imaginary quadratic field guarantees the existence of a curve over $GF(q)$ with order $\#\mathrm{E}(GF(q))$. In Chapter 4, we will discuss the implementation aspects of this result to construct elliptic curves which will be useful for cryptography.

Now, we focus upon elliptic curves over finite field which will be of interest to us from the point of designing a public key cryptosystem.

2.5 Elliptic Curves over Finite Fields

So far, our study of elliptic curves has been limited to complex analysis, algebraic number theory and modular forms. None of these gives a reason for why Weierstrass elliptic equation is called an elliptic *curve*. Now we will study the Weierstrass elliptic equation as a curve in algebraic geometry. Please see Appendix D for introductory overview of some of the basic concepts of algebraic geometry which will be required in the sequel.

From algebraic geometric viewpoint, elliptic curves are algebraic varieties of genus one and are given by the general form of Weierstrass elliptic equation over any field K.

$$f : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad for\ a_i \in \mathrm{K}. \tag{2.3}$$

Here, K is a finite or algebraically closed field. The corresponding equation in projective space can be obtained by substituting $x = X/Z$ and $y = Y/Z$

$$E : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3, \quad for\ a_i \in \mathrm{K}. \tag{2.4}$$

This homogeneous equation in projective plane is important as it also helps in defining an extra point on elliptic curve which is not visible in affine plane. This point is called "point at infinity" and is the only point with 0 Z-coordinate, namely (0,1,0). The set of all the points in affine plane along with this special point is called elliptic curve.

Since, in this section our major concern is with elliptic curves over $GF(q)$, we will consider K to be $GF(q)$. Since any cubic curve can have at the most $(3-1)(3-2)/2 = 1$ double point (See Appendix D), the genus 1 of elliptic curves implies that they are non-singular. Hence, any Weierstrass equation should be such that $\partial f/\partial X, \partial f/\partial Y$ and $\partial f/\partial Z$ are not zero simultaneously at any point on the curve. An equivalent definition of non-singularity of an elliptic curve is given by its discriminant. Any Weierstarss equation with non zero discriminant will be non-singular. Let us first define various parameters for Weierstrass elliptic equation given above.

$$\begin{aligned} c_4 &= (a_1^2 + 4a_2)^2 - 24(2a_4 + a_1a_3), \\ c_6 &= (a_1^2 + 4a_2)^3 + 36(a_1^2 + 4a_2)(2a_4 + a_1a_3) - 216(a_3^2 + 4a_3), \\ \Delta &= (c_4^3 - c_6^2)/1728, \qquad (2.5) \\ j(E) &= 1728c_4^3/\Delta. \qquad (2.6) \end{aligned}$$

Two elliptic curves are said to be isomorphic if they are isomorphic as projective varieties. As we know from the earlier discussion that two elliptic curves over a finite field are isomorphic if their j-invariants and orders are equal. Hence, isomorphism of projective varieties of two elliptic curves can be expressed in terms their j-invariant and order $\#\mathrm{E}(GF(q))$. Equivalently, any birationally equivalent transformation of Weierstarss equation will give

another isomorphic elliptic curve. Obviously, j-invariant and order of the elliptic curve will be invariant under such birational transformation. The following result relates the notion of isomorphism of elliptic curves to the coefficients of Weierstrass equations that defines the curves [Sil85, Men93b].

Theorem 2.9 *Two elliptic curves $E_1(GF(q))$ and $E_2(GF(q))$ given by equations*

$$\begin{aligned} E_1 : y^2 + a_1xy + a_3y &= x^3 + a_2x^2 + a_4x + a_6 \\ E_2 : y^2 + \overline{a_1}xy + \overline{a_3}y &= x^3 + \overline{a_2}x^2 + \overline{a_4}x + \overline{a_6} \end{aligned}$$

are isomorphic over $GF(q)$ iff there exists a birational transformation (i.e. change of variables)

$$(x, y) \longrightarrow (u^2x + r, u^3y + u^2sx + t)$$

which transforms equation E_1 to equation E_2.

Using this theorem, it is possible to transform the Weierstrass elliptic curve equation to a considerably simpler form. It can be easily shown that

1. if $char(GF(q)) \neq 2$ then Weierstrass equation can be transformed by change of variable $(x, y) \longrightarrow (x, y - \frac{a_1}{2}x - \frac{a_6}{2})$ into another isomorphic form $E' : y^2 = x^3 + b_2x^2 + b_4x + b_6$,
2. if $char(GF(q)) \neq 2, 3$ then Weierstrass equation can further be transformed by change of variables $(x, y) \longrightarrow (x, y - \frac{a_1}{2}x - \frac{a_6}{2})$ into another isomorphic form $E'' : y^2 = x^3 + ax + b$,
3. if $char(GF(q)) = 2$ then there will be two cases
   (a) if $j(E) \neq 0$ then Weierstrass equation can be reduced to $E' : y^2 + xy = x^3 + ax^2 + b$,
   (b) if $j(E) = 0$ then $E' : y^2 + ay = x^3 + bx + c$.

Having seen the isomorphic transformations, now we discuss the group structure of the set of all points of an elliptic curve over a finite field $GF(q)$.

2.6 Group Law

The set of all the solutions of Weierstrass elliptic curve equation (and also the isomorphic reduced forms) takes a structure of an abelian group in a natural way. Though the structure of the group is unique but there are several ways to define it. As we saw in Section 2.1, that the set of solutions of Weierstrass equation over $\mathbb{C}$ is isomorphic to complex lie group $\mathbb{C}/\Lambda$, it can be shown that reduction map reduces this group to a group of finite torsion points.

From algebraic geometric point of view, elliptic curves are described as an abelian variety of genus one. The Picard group [Sil85](also see Appendix D) of any smooth algebraic curve (an algebraic variety of dimension 1) is given by set of equivalence classes of 0 degree divisors under modulo action of set principal divisors. In this section we will see that, in case of elliptic curves, each element of Picard group (i.e. a divisor of degree 0) can be represented by a point on curve. Hence, by defining a binary law for points on elliptic curve analogous to that for picard group, the elliptic curve variety can be given a group structure. Here, we define the binary law in a rather different and simpler manner.

Let E be elliptic curve group over a finite field $GF(q)$(not necessarily) together with the point at infinity $\mathcal{O}$ as defined above. Since Weierstrass equation has degree 3, any line in affine plane will intersect the curve at exactly three points, say $\mathcal{P}, \mathcal{Q}, \mathcal{R}$. Note if the line is tangent then $\mathcal{P}, \mathcal{Q}, \mathcal{R}$ may not be distinct. In such a case, point considered with proper multiplicities will give count 3. Now, we define the composition law, $\oplus$ referred as addition, by the following rule.

Composition Law: Let $\mathcal{P}, \mathcal{Q} \in$E, L the line connecting $\mathcal{P}$ and $\mathcal{Q}$ (tangent if $\mathcal{P} = \mathcal{Q}$), and $\mathcal{R}$ the third point of intersection of L with E. Let L$'$ be the line connecting $\mathcal{R}$ and $\mathcal{O}$. Then $\mathcal{P} \oplus \mathcal{Q}$ is the point such that L$'$ intersects E at $\mathcal{R}, \mathcal{O}$, and $\mathcal{P} \oplus \mathcal{Q}$.

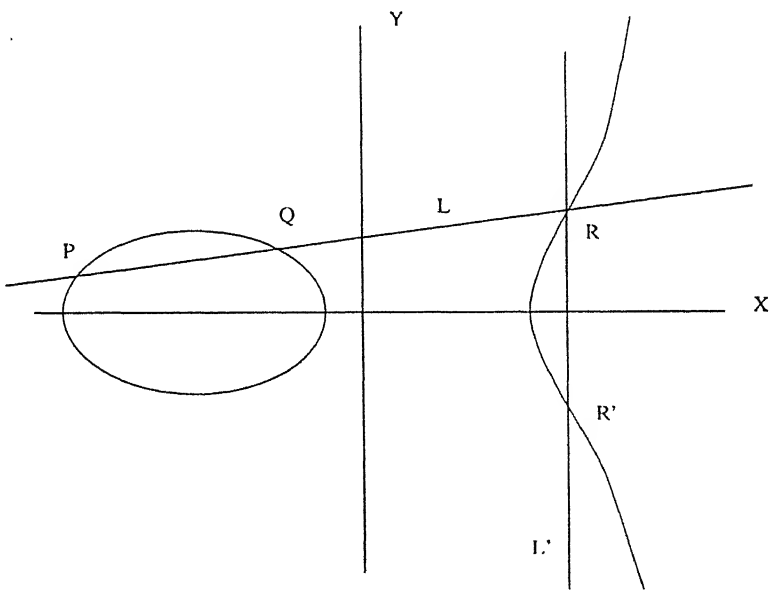


Figure 2.1: Elliptic Curve over $\mathbb{R}$

This composition law can easily be verified to hold all the axioms required for an abelian group.

Proposition 2.6 *1. If a line intersects E at (not necessarily distinct) points $\mathcal{P}, \mathcal{Q}, \mathcal{R}$, then $(\mathcal{P} \oplus \mathcal{Q}) \oplus \mathcal{R} = \mathcal{O}$.*

2. $\mathcal{P} \oplus \mathcal{O} = \mathcal{P}$ *for all* $\mathcal{P} \in E$.

3. $\mathcal{P} \oplus \mathcal{Q} = \mathcal{Q} \oplus \mathcal{P}$ *for all* $\mathcal{P}, \mathcal{Q} \in E$.

4. *Let* $\mathcal{P} \in E$. *There is a point of* E, *denoted by* $\ominus\mathcal{P}$, *such that* $\mathcal{P} \oplus (\ominus\mathcal{P}) = \mathcal{O}$.

5. *Let* $\mathcal{P}, \mathcal{Q}, \mathcal{R} \in E$. *then* $(\mathcal{P} \oplus \mathcal{Q}) \oplus \mathcal{R} = \mathcal{P} \oplus (\mathcal{Q} \oplus \mathcal{R})$.

Proof of all, except the (5) (associativity), is quite immediate. For (5), [Cha88] contains an interesting proof based on Bezout's theorem.

Hence we see that the addition of two points on an elliptic curve is the inverse of the point at which the line passing through those two points cuts the elliptic curve. Now we will derive explicit expressions for the addition of two points to obtain the coordinates of the third point. Let $\mathcal{P}, \mathcal{Q}, \mathcal{R} \in$E and $\mathcal{O}$ be the point at infinity (0,1,0). Moreover, $\mathcal{P} = (x_1, y_1)$, $\mathcal{Q} = (x_2, y_2)$ and $\mathcal{R} = (x_3, y_3)$. The corresponding projective coordinates are given by $\mathcal{P} = (X_1, Y_1, Z_1)$, $\mathcal{Q} = (X_2, Y_2, Z_2)$ and $\mathcal{R} = (X_3, Y_3, Z_3)$. The affine point corresponding to a projective point is obtained by transformation $x_i = X_i/Z_i$ and $y_i = Y_i/Z_i$. Let $l : y = mx + c$ be the line passing through the points $\mathcal{P}$ and $\mathcal{Q}$.

Now for general form of Weierstrass equation, the slope of the line will be given by

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & if\ \mathcal{P} \neq \mathcal{Q}, \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}, & if\ \mathcal{P} = \mathcal{Q}. \end{cases}$$

and $c = y_1 - mx_1$. Now by simple algebraic manipulation, the coordinates of third point of intersection of line l and elliptic curve can be obtained. If the $\mathcal{P} \oplus \mathcal{Q} = \mathcal{R}$, then the coordinates of third point will be $(x_3, -y_3)$. Hence

$$x_3 = m^2 + a_1m - a_2 - x_1 - x_2, \tag{2.7}$$

$$y_3 = -(m + a_1)x_3 - c - a_3. \tag{2.8}$$

From these equations, we can see that sum of two points in E can easily be computed. The next theorem tells us that this composition law is same as that for the Picard group [Sil85, TV91].

Theorem 2.10 *Let* E *be an elliptic curve.*

(a) For every divisor $Div^0(E)$ *there exist a unique point* $\mathcal{P} \in E$ *so that* $D \sim (\mathcal{P}) - (\mathcal{O})$.
Let

$$\sigma : Div^0(E) \longrightarrow E$$

be the map given by this association.

(b) The map σ is surjective.

(c) Let $D_1, D_2 \in Div^0(E)$. then

$$\sigma(D_1) = \sigma(D_2) \qquad iff \quad D_1 \sim D_2.$$

Thus σ induces a bijection of sets

$$\sigma : Pic^0 \cong E.$$

If E is given by a Weierstrass equation, then the group law defined above and the group law induced from $pic^0(E)$ by using σ are the same.

Please refer to [Sil85, TV91] for the proof. There is another way of defining the binary composition law for elliptic curves using division polynomials [Kob90, Sch85, Men93b].

The next theorem tells us about bound on the order of the elliptic curve over $GF(q)$ [Sil85].

Theorem 2.11 (Hasse Theorem) *Let $\#E(GF(q))= q + 1 - t$. Then $|t| \leq 2\sqrt{q}$.*

An elliptic curve is said to be *supersingular* if t is divisible by the characteristic of $GF(q)$. Otherwise it is called non-supersingular. In fact, an elliptic curve is supersingular if and only if $t^2 = 0, q, 2q, 3q,$ *or* $4q$. As discussed in Section 2.3, the endomorphism ring of a supersingular elliptic curve is isomorphic to an order in quaternion algebra whereas that of a non-supersingular elliptic curve is isomorphic to an order in an imaginary quadratic field. From this definition of supersingular elliptic curves, it is easy to observe that the order of any non-supersingular elliptic curve over $GF(2^n)$ will always be even as t is coprime to 2, the characteristic of field $GF(2^n)$. The proof of the Hasse theorem is obvious for non-supersingular elliptic curves from Section 2.4 as every non-supersingular elliptic curve satisfies the two norm equations for some proper discriminant. We will see in Chapter 4 that these two norm equation can be written as

$$\begin{aligned} q &= x^2 + dy^2 \\ \#E(GF(q)) &= q + 1 - 2x \end{aligned}$$

where $2x, 2y \in \mathbb{Z}$. From first equation we see that $|2x| \leq 2\sqrt{q}$. Substituting this in the second equation, we get the bound given by Hasse Theorem. The Bound given in Hasse Theorem is valid for supersingular elliptic curves as well. For a detailed proof please refer to [Sil85].

From the theory of abelian groups, we know that any finite abelian group G can be decomposed into a direct sum of cyclic groups

$$G = \mathbb{Z}_{n_1} \otimes, \ldots, \otimes \mathbb{Z}_{n_t}$$

where $n_{i+1} | n_i$. We say that the group G is an abelian group of type $(n_1, \ldots, n_t)$ and rank s. The next theorem tells us about the structure of the elliptic curve group.

Theorem 2.12 *$E(GF(q))$ is an abelian group of rank 1 or 2. The type of the group is (n_1, n_2), i.e. $E(GF(q)) \cong \mathbb{Z}_{n_1} \otimes \mathbb{Z}_{n_2}$, where $n_2 | n_1$ and furthermore $n_2 | q - 1$.*

The elliptic curve group will be cyclic if its rank is one or equivalently, if n_2 and $q - 1$ are relatively prime.

With this we conclude this chapter. In Chapter 3, we will discuss the discrete logarithm problem in elliptic curve group and its application in public key cryptography.

Chapter 3

Elliptic Curve Discrete Logarithm Problem and Cryptosystems

In Chapter 2, we began with Weierstrass elliptic function $\wp(z)$ and later obtained the non-supersingular elliptic curve over finite field $GF(q)$ using reduction maps. Now, we discuss the elliptic curve over $GF(q)$ so as to understand their use in designing an efficient and secure cryptosystem as mentioned in Chapter 1. We will also introduce the **Discrete Logarithm Problem** and briefly survey various algorithms known for solving it. Since, the intractability of discrete logarithm problem (**DLP**) in any finite abelian group G is a measure of the security of cryptosystem defined over G, our aim will be to determine all conditions for which known algorithm will fail. In the first section, we give the point addition formulae for elliptic curve in $GF(p)$ and $GF(2^n)$ which are useful from the point of view of implementation.

3.1 Elliptic Curves over $GF(2^n)$ and $GF(p)$

As we have seen that the addition of two points in elliptic curve group requires arithmetic in underlying field. From the practical point of view, finite fields $GF(p)$ and $GF(2^n)$ are of interest as their arithmetic is simple and more suitable for hardware and software implementation. In this section, we will concentrate on elliptic curves over these fields only.

The finite field $GF(2^n)$ is more attractive for hardware implementation as its elements are represented by a string of n bits. Moreover, addition in this field is equivalent to XORing of binary bits and multiplication can be done by a logic circuit [LN94, Men93a, McE87]. We will discuss the efficient arithmetic related issues in Chapter 6 in detail. Here, we give the addition formulae for elliptic curves over $GF(2^n)$.

As we discussed in Chapter 2, the Weierstrass elliptic curve equation over $GF(2^n)$ can be reduced to an isomorphic form. For supersingular elliptic curves the reduced equation is

as given below [Men93a, Men93b, AMV93].

$$y^2 + ay = x^3 + bx + c \qquad where \;\; a, b, c \in GF(2^n)$$

It can easily be shown that the j-invariant for these curve will always be zero. We will see in Chapter 4 that there are only 3 and 7 isomorphism classes of supersingular curves for n odd and even respectively. Now, we define the addition for supersingular elliptic curves. Let $\mathcal{P}= (x_1, x_2) \in$E; then $\ominus\mathcal{P}= (x_1, y_1+a)$. If $\mathcal{Q}= (x_2, y_2) \in$E and $\mathcal{Q} \neq \ominus\mathcal{P}$, then $\mathcal{P} \oplus \mathcal{Q}= (x_3, y_3)$, where

$$x_3 = \begin{cases} (\frac{y_1+y_2}{x_1+x_2})^2 + x_1 + x_2 & \mathcal{Q} \neq \mathcal{P} \\ \frac{x_1^4+b^2}{a^2} & \mathcal{Q} = \mathcal{P} \end{cases}^{j}$$

and

$$y_3 = \begin{cases} (\frac{y_1+y_2}{x_1+x_2})(x_3 + x_1) + y_1 + a & \mathcal{Q} \neq \mathcal{P} \\ (\frac{x_1^2+b}{a})(x_3 + x_1) + y_1 + a & \mathcal{Q} = \mathcal{P} \end{cases}$$

Notice that addition of two points require 2 multiplications and 1 inverse whereas doubling requires computation of 1 multiplication and inverse of a which can be precomputed.

The reduced curve equation for non-supersingular elliptic curves is given by

$$y^2 + xy = x^3 + ax^2 + b \qquad where \;\; a, b \in GF(2^n)$$

Now, we define the addition for non-supersingular elliptic curves. Let $\mathcal{P}= (x_1, x_2) \in$E; then $\ominus\mathcal{P}= (x_1, x_1 + y_1)$. If $\mathcal{Q}= (x_2, y_2) \in$E and $\mathcal{Q} \neq \ominus\mathcal{P}$ then $\mathcal{P} \oplus \mathcal{Q}= (x_3, y_3)$, where

$$x_3 = \begin{cases} (\frac{y_1+y_2}{x_1+x_2})^2 + (\frac{y_1+y_2}{x_1+x_2}) + x_1 + x_2 + a & \mathcal{Q} \neq \mathcal{P} \\ x_1^2 + \frac{b}{x_1^2} & \mathcal{Q} = \mathcal{P} \end{cases}$$

and

$$y_3 = \begin{cases} (\frac{y_1+y_2}{x_1+x_2})(x_3 + x_1) + y_1 + x_3 & \mathcal{Q} \neq \mathcal{P} \\ x_1^2 + (x_1 + \frac{y_1}{x_1})x_3 + x_3 & \mathcal{Q} = \mathcal{P} \end{cases}$$

In this case, the point addition requires 2 multiplications and 1 inverse and doubling requires 3 multiplications and 1 inverse. Hence, from implementation point of view, supersingular curves are more efficient. However, as but we will see that there are only limited number of choices for supersingular elliptic curves over $GF(2^n)$.

In case of elliptic curve over $GF(p)$ $(p > 3)$ the reduced form is given by

$$y^2 = x^3 + ax + b \qquad where \;\; a, b \in GF(p).$$

This equation is also called short normal form of Weierstrass equation. The j-invariant and discriminant for this equation are given by

$$\triangle = -16(4a^3 + 27b^2) \quad and \quad j = -1728(4a)^3/\triangle$$

Here addition is defined as follows:
Let $\mathcal{P}= (x_1, x_2) \in$E; then $\ominus\mathcal{P}= (x_1, -y_1)$. If $\mathcal{Q}= (x_2, y_2) \in$E and $\mathcal{Q} \neq \ominus\mathcal{P}$ then $\mathcal{P}\oplus\mathcal{Q}= (x_3, y_3)$, where

$$x_3 = m^2 - x_1 - x_2, \tag{3.1}$$

$$y_3 = m(x_1 - x_3) - y_1, \tag{3.2}$$

with

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & if\ \mathcal{P} \neq \mathcal{Q}, \\ \frac{3x_1^2 + a}{2y_1}, & if\ \mathcal{P} = \mathcal{Q}. \end{cases}$$

We will see in Chapter 6 that inverse computation can be avoided by using projective coordinate. This leads to reduction in computations needed to be carried out.

In the next section, we explain the discrete logarithm problem in finite abelian groups and also in elliptic curve group.

3.2 Discrete Logarithm Problem

The discrete logarithm problem (DLP) in a finite multiplicative group G refers to computation of x for two element a, b of G such that $a^x = b$. The integer x is said to logarithm of b to base a. This problem is known to be very difficult if the group order is large. The intractability of this problem motivated Diffie and Hellman to introduce the concept of public key cryptography which exploits the difficulty of the finding discrete logarithm in a finite group as a security measure. Initially, Diffie and Hellman's idea was limited to discrete logarithm problem in multiplicative group of $GF(p)$ but in 1985 it was generalized by ElGamal [ElG85] for any finite abelian group. The application of DLP in cryptography has been one of the reasons for increased attention towards solving this problem. Consequently, several algorithm and techniques have been proposed in the recent past for finding logarithms in finite abelian groups.

Under the threat posed by the ongoing research in this area, the search for small groups with relatively difficult discrete logarithm problem has been of great interest for cryptographers. The elliptic curve emerges in cryptography as one of the outcome of this search. The discrete logarithm problem in an elliptic curve group E refers to computation of k for given $\mathcal{P}, \mathcal{Q} \in$E such that

$$\mathcal{Q} = k\mathcal{P} = \overbrace{\mathcal{P} \oplus \mathcal{P} \oplus \ldots \oplus \mathcal{P}}^{k \text{ times}}$$

Here, we first briefly survey various algorithms for finding discrete logarithm in a finite abelian group. In each case, we will also discuss the the complexity of discrete logarithm

problem in elliptic curve group. An elliptic curve group can either be cyclic or type (n_1, n_2) with rank 2. If elliptic curve group is not cyclic then we will concentrate on its subgroup of order n_2 which will be cyclic. It should be noted that for a non cyclic elliptic curve of type (n_1, n_2), integer n_1 divides n_2. This survey will also help us in finding the constraints on the selection of a group for cryptosystem so as to frustrate all known algorithms. We will discuss the public key algorithms for elliptic curve cryptosystems in Section 3.4

The algorithms which are known for finding discrete logarithms in a finite abelian group can be categorized as follows. In the discussion given below, the groups are assumed to be abelian.

1. The algorithms which works in any arbitrary cyclic group (square root attacks).
2. The algorithm which works in any arbitrary group but exploits the subgroup structure (Pohlig-Hellman Method [PH78]).
3. Index Calculus Method.
4. MOV reduction attack for elliptic curves.

Now, we discuss each of these case separately.

3.2.1 Square Root Methods

In this section, we will discuss the algorithms in arbitrary cyclic groups. These algorithms are called square root method because their computational complexity is of the order of square root of the size of the group. Unless otherwise specified, G will denote a finite abelian group of order m with α as the generator element of the group. Let $m' = \lceil \sqrt{m} \rceil$

Baby-Step Giant Step Method

This method is due to Shanks [Sch85]. Let $\alpha^x = \beta$. Hence, discrete logarithm problem refers to computation of $x = \log_\alpha \beta$. This methods begin with precomputing a list of pairs (i, α^i) for $0 \leq i \leq m'$ and storing it in memory. The stored list is sorted by second component. Now for each j, $0 \leq j \leq m'$, compute $\beta\alpha^{-jm'}$ and check by applying binary search whether it equals any of α^i in the stored list. If the match is found for some i and j then

$$\begin{aligned} \beta\alpha^{-jm'} &= \alpha^i \\ \Rightarrow \quad \beta &= \alpha^{i+jm'} \\ \Rightarrow \quad \log_\alpha \beta &= i + jm'. \end{aligned}$$

This algorithm requires memory for storing $O(m')$ entries of table. Moreover, $O(m')$ search operations are to be performed where each binary search requires $O(\log m')$ steps.

Hence, total number of steps involved are $O(m' \log m')$. Apart from this, $\beta\alpha^{-jm'}$ needs to be computed before each of $O(m')$ search which requires two multiplications (α^{-1} needs to be computed in the beginning only and rest of the terms can be computed by repeated multiplication). Hence, we see this algorithm requires storage of m' entries and $O(m'(\log m' + c))$ steps. In case of elliptic curves, the list will consists of multiples of generator point which further increases the storage requirements. In fact, instead of storing both the coordinates of a point, only the part of x-coordinate can be stored. This reduces the storage requirements significantly.

Now, if we select a finite abelian group (any group, not necessarily elliptic curve group) such that its order is at least 10^{30}, then this algorithm will not be good from point of view of actual implementation. Hence, we get a restriction on the group to be selected for cryptosystem that its order should be greater than 10^{30}.

Pollard ρ-Method

In [Pol78], Pollard discusses a probabilistic algorithm which does not require the precomputation of list of logarithms. By some heuristic arguments, it is shown that this algorithm requires $O(m')$ steps. Hence, this will also fail if the order of the group is very large. Please refer [Men93a, Pol78] for detail.

3.2.2 Pohlig-Hellman Method

This method for computing discrete logarithm exploits the subgroup structure of the group G. It requires factorization of m to determine various subgroups of G and then computes the discrete logarithm problem in each of the subgroup by using square root method. Final result is obtained by applying Chinese remainder theorem [IR82]. Let

$$m = \prod_{i=1}^{t} p_i^{e_i}$$

where p_i are prime numbers and e_i are positive exponents. The algorithm begins with computation of $z_i = x \ mod \ p^{e_i}$ for each i. Here $x = \log_\alpha \beta$.

Suppose that $z_i = \sum_{j=0}^{e_i-1} z_{ij} p_i^j$ where $0 \leq z_{ij} < p_i$.
Let γ_i be the p_i^{th} root of unity in G, i.e. $\gamma_i = \alpha^{m/p_i}$. Then

$$\begin{aligned} \beta^{m/p_i} &= \alpha^{xm/p_i} \\ &= \gamma_i^{\sum_{j=0}^{e_i-1} z_{ij} p_i^j} \\ &= \gamma_i^{z_{i0}} \end{aligned}$$

Hence, now we need to compute the z_{i0}. Since z_{i0} is logarithm of $\gamma_i^{z_{i0}}$ to the base γ_i in the

cyclic subgroup of order p_i in G, it can be computed using Baby-Step Giant-Step method in $O(\sqrt{p_i}(\log p_i + c))$ steps and with $O(\sqrt{p_i})$ entries. For the computation of z_{i1}, we see that

$$\begin{aligned}(\beta\alpha^{-z_{i0}})^{m/p_i^2} &= (\alpha^{\sum_{j=1}^{e_i-1} z_{ij}p_i^j})^{m/p_i^2} \\ &= \gamma_i^{z_{i1}}\end{aligned}$$

Thus we can compute z_{i1} and hence other z_{ij} as well for all values of j. Repeating this procedure for each i, z_i can be computed. Once all the z_i are known, the discrete logarithm x can be computed using Chinese remainder theorem.

To determine the complexity of this algorithm, we see that for each p_i, e_i discrete logarithms need to be computed. Hence, this method requires storage of $O(\sum_{i=1}^{t} \log p_i)$ elements and $O(\sum_{i=1}^{t} e_i(\log m + \sqrt{p_i}\log p_i))$ steps for computation of discrete logarithm.

To make this attack infeasible, it is necessary that the m should contain a large prime factor. The prime factor must be greater than 30 decimal digits integer.

3.2.3 Index Calculus Method

This method is considered to be most powerful for computing logarithms in finite abelian groups but it does not apply to any arbitrary abelian finite group. We will see that this method successfully applies over multiplicative group in $GF(p)$ and $GF(2^n)$, whereas in case of elliptic curve group it is not yet shown to be applicable. This justifies the superiority of elliptic curve cryptosystems over other cryptosystem based on discrete logarithm problem in finite fields. A brief description of the algorithm follows [Sim91, Men93a].

The standard index calculus approach consists of two stages. The first is the precomputation (carried out once for a given group) of a large subset $S = \{\gamma_1, \ldots, \gamma_n\}$ of G with the property that a significantly large fraction of G can be expressed as product of elements of S. The set S is called *factor base*. Now for computation of logarithms of the elements of factor base with respect to α, approximately n $(=|S|)$ linear equation relating n unknowns, the unknowns being the required logarithms, are obtained. The logarithms, then, are obtained by solving by these equation. For obtaining linear equations, we choose an integer a randomly and try to express it as the product of γ_i.

$$\alpha^a = \prod_{i=1}^{n} \gamma_i^{c_i}$$

In case of failure the process is repeated, otherwise we obtain

$$a = \sum_{i=1}^{n} c_i \log_\alpha \gamma_i \ (mod\ m)$$

After collecting sufficient number of such equations, we solve them for $\log_\alpha \gamma_i$. This completes the computation of the first stage.

Now for computing $x = \log_\alpha \beta$, we try to express $\alpha^s\beta$ in terms of γ_i for a randomly chosen s.

$$\alpha^s\beta = \prod_{i=1}^{n} \gamma_i^{c_i}$$

If we succeed in this process then

$$x = \log_\alpha \beta = \sum_{i=1}^{n} e_i \log_\alpha \gamma_i - s \ (mod\ m)$$

The complexity of this method is given by

$$L[m, a, c] = O(exp((c + O(1))(\log m)^a(\log\log m)^{1-a}))$$

where c is a constant and $0 \leq a \leq 1$. The complexity of this algorithm is largely dependent upon the problem of selecting minimal factor base which spans a large fraction of G. For prime field $GF(p)$, the obvious choice of factor base is set of sufficient number of prime integers and for $GF(2^n)$ set of all the polynomials of degree less than some appropriately chosen integer d [Sim91]. This algorithm poses a serious threat to cryptosystems which are based on the discrete logarithm problem in $GF(p)$ and $GF(2^n)$. The minimum size of the field for a cryptosystem to be secure under this attack is approximately 2^{700}. Whereas in case of elliptic curve, there seems to be no proper choice for the factor base S. The most natural seems to be the point of small height[1] in E($\mathbb{Q}$) [Sil85]. But there are few points of small height and it is very difficult to find them. Even if such a set is found, finding a practical method for lifting any point in E($GF(q)$) to a point in E($\mathbb{Q}$) is not known [Men93b, Sil85].

Now we discuss the MOV (Menezes,Okamato,Vanstone) [Men93a, Men93b] attack which specifically works on elliptic curves only.

3.2.4 MOV Reduction Attack

This method reduces the discrete logarithm problem in elliptic curve group $GF(q)$ to the discrete logarithm problem in a suitable finite extension $GF(q^k)$ of $GF(q)$ using Weil Pairing method [Men93a, Men93b, Sil85]. This is achieved by establishing an isomorphism between $< \mathcal{P} >$, the subgroup of E($GF(q)$) generated by $\mathcal{P}$, and the subgroup of same order of a suitable finite extension $GF(q)$. Hence for this attack to be applicable, the multiplicative group of extension field $GF(q^k)$ must be divisible by #E($GF(q)$) (=m). That means

$$q^k \equiv 1 \ mod\ m. \tag{3.3}$$

[1] height function on any point on elliptic curve is the discrete logarithm of x-coordinate. In a way, it also means the numbers of decimal digits or bits in x-coordinate

Let us first define Weil pairing [Sil85, Men93a, Men93b]. Let l be a positive integer relatively prime to q. Then the Weil pairing e_l is a function

$$e_l : E[l] \times E[l] \longrightarrow GF(q^k)$$

where $E[l]$ is subgroup of elliptic curve group of order l.

Now we give some of the important properties of the Weil pairing function e_l.

1. Identity: For all $\mathcal{P} \in$E[l], $e_l(\mathcal{P}, \mathcal{P})= 1$.

2. Alteration: For all $\mathcal{P}, \mathcal{Q} \in$E[$l$], $e_l(\mathcal{P}, \mathcal{Q})= e_l(\mathcal{Q}, \mathcal{P})^{-1}$.

3. Bilinearity: For all $\mathcal{P}, \mathcal{Q}, \mathcal{R} \in$E[$l$]

$$\begin{aligned} & e_l(\mathcal{P} \oplus \mathcal{Q}, \mathcal{R}) &= e_l(\mathcal{P}, \mathcal{R}) e_l(\mathcal{Q}, \mathcal{R}) \\ \textit{and} \quad & e_l(\mathcal{P}, \mathcal{Q} \oplus \mathcal{R}) &= e_l(\mathcal{P}, \mathcal{Q}) e_l(\mathcal{P}, \mathcal{R}) \end{aligned}$$

4. If E[l] $\subseteq$E($GF(q)$), then $e_l(\mathcal{P}, \mathcal{Q}) \in GF(q)$ for all $\mathcal{P}, \mathcal{Q} \in$E[$l$].

The Weil pairing function can be obtained in terms of function of principal divisors [Sil85, Men93a, Men93b] of $\mathcal{P}$ and $\mathcal{Q}$. Please see [Men93a, Men93b] for further details.

Let us assume that E[l] $\subseteq$E($GF(q)$) and also for finite extension $GF(q^k)$ of $GF(q)$ E[l] $\subseteq$E($GF(q^k)$). Next theorem defines the required isomorphism.

Theorem 3.1 *Let $f :< \mathcal{P} > \longrightarrow \mu_l$ be defined by $f : \mathcal{R} \longmapsto e_l(\mathcal{R}, \mathcal{Q})$. Then f is a group isomorphism. Here $\mathcal{P}, \mathcal{Q}, \mathcal{R} \in E[l]$ and $\mu_l \in GF(q^k)$.*

Now, let $\mathcal{P}$ be a point in E($GF(q)$) of order l such that $\gcd(q, l) = 1$ and $\mathcal{R} \in < \mathcal{P} >$. Then consider the following algorithm for the computation of s such that $\mathcal{R}= s\mathcal{P}$.

Procedure ECDLP($\mathcal{P}, \mathcal{Q}, \mathcal{R}$,$l$,$GF(q)$)

1. Begin.

2. Find the smallest extension degree k such that E[l] $\subseteq$E($GF(q^k)$).

3. Find $\mathcal{Q} \in$E[l] such that $\alpha = e_l(\mathcal{P}, \mathcal{Q})$ and $\alpha^l = 1$.

4. Compute $\beta = e_l(\mathcal{R}, \mathcal{Q})$.

5. Compute s, the discrete logarithm of β to the base α in $GF(q^k)$.

6. End.

The correctness of the algorithm is immediate as

$$\begin{aligned} \beta &= e_l(\mathcal{R}, \mathcal{Q}) \\ &= e_l(s\mathcal{P}, \mathcal{Q}) \\ &= e_l(\mathcal{P}, \mathcal{Q})^s \qquad \text{(Bilinearity)} \\ &= \alpha^s \end{aligned}$$

This algorithm requires the minimum field extension in which E[l] can be embedded. If k is large then the algorithm takes long time in finding discrete logarithm as the size of extension field becomes very large. Since there are few isomorphism classes of supersingular elliptic curves (see Chapter 4) and for each class the order of the curve is known, the extension degree can easily be found. It is found that the maximum extension degree for minimal suitable extension of working field is 4 and 6 for $GF(2^n)$ and $GF(p)$ respectively [Men93a, Men93b]. Hence this method is very efficient for supersingular elliptic curves.

In case of non-supersingular elliptic curves there are plenty of choices for the order of the curve over a given finite field. Hence, the elliptic curve can be so selected that MOV attack becomes infeasible. If the order of the curve is not relatively prime to characteristic of field then Equation 3.3 will not hold for any value of k. If an elliptic curve is defined over $GF(p)$ such that its order is p then MOV attack will not be applicable over this elliptic curve. Hence such curves will be of great interest in cryptography. In Chapter 7, we will give examples of such curves.

The order of non-supersingular elliptic curves over $GF(2^n)$ is always even and hence is not relatively prime to the characteristic of the field $GF(2^n)$. But in this case the above algorithm can be applied with a minor modification. In [Miy91], Miyaji discusses a variation of the above algorithm to reduce the discrete logarithm problem in $E(GF(2^n))$ to discrete logarithm problem in some suitable extension of $GF(2^n)$.

Let $\#E(GF(2^n))= m = c*q$ and $\mathcal{P}, \mathcal{R} \in E(GF(2^n))$ such that $\mathcal{R} = s\mathcal{P}$. Here c is the even factor of the order of the curve and q is the prime. Further assume $\mathcal{P}' = c\mathcal{P}$ and $\mathcal{R}' = c\mathcal{R}$. Notice that $\mathcal{R}'$ will lie in $< \mathcal{P}' >$ because $\mathcal{R} \in < \mathcal{P} >$. Since the order of $< \mathcal{P}' >$ is q, the MOV reduction algorithm can be applied to find an integer s' such that $\mathcal{R}' = s'\mathcal{P}'$. The point $\mathcal{R} - s'\mathcal{P}$ will definitely belong to $< q\mathcal{P} >$ because $\mathcal{R}' - s'\mathcal{P}' = c(\mathcal{R} - s'\mathcal{P})$ is point at infinity. Now we compute $s'\mathcal{P}$ and find an integer s'' such that $\mathcal{R} - s'\mathcal{P} = s''(q\mathcal{P})$. Since the order of $< q\mathcal{P} >$ is c, integer s'' can easily be computed. Once s' and s'' are known, the integer s can be computed easily by setting $s \equiv s' + s''q \ mod \ m$. This discussion completes the proof of the following.

Theorem 3.2 *For any elliptic curve $E(GF(2^n))$ and any point $\mathcal{P} \in E(GF(2^n))$, we can reduce the discrete logarithm problem in elliptic curve group to another elliptic curve group whose order is either equal or less than the original group.*

In the next section we summarize the preceding discussion with restriction on the selection of elliptic curves such that all of above algorithms fail.

3.3 Restrictions on Selection of Elliptic Curve

As we discussed in the previous section, the order of the elliptic curve group must be greater than a 30 digit decimal number to make the square root attacks infeasible. Moreover, if the order of the curve contains a large prime factor then the Pohlig-Hellman attack will be no better than square root attacks. Hence, from the point of view of security, the elliptic curve should be selected such that its order contains a prime factor greater than 30 digit decimal number.

The index calculus is considered to be impractical over elliptic curve group but surely is a serious threat for cryptosystems based on discrete logarithm problem in $GF(2^n)$ and $GF(p)$.

Now, we discuss the restrictions on selection of elliptic curve to make the MOV reduction attack infeasible [CTT94]. As we discussed in the previous section that the MOV attack is based on isomorphism of $\mathrm{E}(GF(q))$ with a subgroup in some suitable extension field $GF(q^k)$. Let us assume that $\#\mathrm{E}(GF(q))= m = c * q'$ where c is small number (say less than 100) and q' is the large prime factor (greater 30 decimal digits).

For curves over $GF(p)$

Here $q = p$. If the extension field $GF(p^k)$ contains a subgroup isomorphic to $\#\mathrm{E}(GF(p))$, then

$$p^k \equiv 1 \ mod \ m$$

Since $\gcd(p, m) = 1$, from Euler's generalization of Fermat's theorem we can say that k must be a factor of $\varphi(m)$, where φ is Euler's totient function. First we make following definition.

Definition 3.1 *A number x is said to be B-nonsmooth if it has no factor less than B.*

Now, if we select the prime factor q' such that $(q-1)/2$ is B-nonsmooth and

$$p^{2\varphi(c)} \not\equiv 1 \ mod \ m,$$

then we can be sure that the minimum extension degree for MOV attack to be applicable will be greater than B. Hence, by taking an appropriately large B, the MOV attack can be frustrated for non-supersingular elliptic curves over $GF(p)$.

For curves over $GF(2^n)$

Here $q = 2^n$. For supersingular elliptic curves the MOV attack is very effective. There are 7 and 3 isomorphism classes of supersingular elliptic curves over $GF(2^n)$ for even and odd n respectively. It is known that maximum value of minimum degree of extension for MOV attack to be applicable is 4. While selecting a curve for cryptosystem, the working field should be so chosen that order of the curve contains a prime factor greater than 30 decimal digits and extension field for MOV attack is larger than $GF(2^{700})$.

Since the non-supersingular elliptic curves over $GF(2^n)$ always have even order, the small factor c of $m = c * q'$ will be even. Hence if the base point $\mathcal{P}$ for $\mathrm{E}(GF(2^n))$ is the generator of the curve then there will be no k for which $q^k \equiv 1 \ mod \ m$ will hold because q and m are not relatively prime. In such a case, Miyaji's variation of MOV algorithm can be used. According to this variation, the discrete logarithm problem is mapped to the discrete logarithm problem in the subgroup of $E(GF(2^n))$ which has an odd order. Let c' be odd part of c. If the prime factor of the order q' is such that $(q'-1)/2$ is B-nonsmooth and

$$q^{2\varphi(c')} \not\equiv 1 \ mod \ (c'q')$$

then the minimum extension degree will be greater than B.

Hence, to ensure the security of the cryptosystem, the elliptic curve should be so selected that its order contains a prime factor greater than 30 digit decimal number and the extension degree for MOV attack to be applicable is controlled by a lower bound.

3.4 Elliptic Curve Public Key Cryptosystems

As we discussed in Chapter 2, an elliptic curve over a finite field $GF(q)$ (over algebraically closed field as well) takes a group structure with a composition law which involves some arithmetic operations in underlying field. Let $\mathrm{E}(GF(q))$ be the elliptic curve and $\mathcal{P} \in \mathrm{E}(GF(q))$. In this group, for any integer k the computation of $k\mathcal{P}$ refers to

$$\mathcal{Q} = k\mathcal{P} = \overbrace{\mathcal{P} \oplus \mathcal{P} \oplus \ldots \oplus \mathcal{P}}^{k \text{ times}}$$

Now, we use an analog of ElGamal's scheme over $GF(q)$ for defining a public key cryptosystem. According to ElGamal's scheme [ElG85] following information will be made public.

- Working field $GF(q)$.
- The elliptic curve equation over $GF(q)$.
- $\#\mathrm{E}(GF(q))$ the order of the elliptic curve.
- Base point $\mathcal{P}$ which is preferably the generator of $\mathrm{E}(GF(q))$.

- The method for obtaining a point in E($GF(q)$) corresponding to message and vice versa.

Now, every user will determine an integer k ($<\#\mathrm{E}(GF(q))$) randomly and compute $k\mathcal{P}$. The integer k will be the private key of the user and $k\mathcal{P}$ will be the public key. Let k_a and k_b be the private keys of user A and B. Hence their public keys will be given by $\mathcal{P}_a = k_a\mathcal{P}$ and $\mathcal{P}_b = k_b\mathcal{P}$ respectively. The user A can communicate to B according to following protocol.

Encryption

- User A finds an integer k ($<\#\mathrm{E}(GF(q))$) randomly and computes $k(\mathcal{P}_b)$.
- User A now takes the message element m and maps it to some point $\mathcal{M}$ and computes $\mathcal{C} = \mathcal{M} \oplus k\mathcal{P}_b$.
- Then A computes $k\mathcal{P}$ and sends to B along with C.

Decryption

- User B computes $k_b(\mathcal{P}_a)$.
- The message is retrieved by $\mathcal{M} = \mathcal{C} \oplus (\ominus k_b(\mathcal{P}_a))$.
- The message m can be obtained by applying the inverse of the message map used in encryption.

A variation of this scheme can be used for digital signature as well.

Encryption with Digital Signature

- User A finds an integer k ($<\#\mathrm{E}(GF(q))$) randomly and computes $(k + k_a)(\mathcal{P}_b)$.
- User A now takes the message element m and maps it to some point $\mathcal{M}$ and computes $\mathcal{C} = \mathcal{M} \oplus (k + k_a)\mathcal{P}_b$.
- Then A computes $k\mathcal{P}$ and sends to B along with C.

Decryption with Signature Verification

- User B computes $\mathcal{P}_a \oplus k\mathcal{P}$.
- Now, B computes $k_b(\mathcal{P}_a \oplus k\mathcal{P})$.

- The message is retrieved by $\mathcal{M}= \mathcal{C} \oplus (\ominus k_b(\mathcal{P}_a \oplus k\mathcal{P}))$.
- The message m can be obtained by applying the inverse of the message map used in encryption.

Here, we see that security of this cryptosystem depends upon the difficulty of computing k from given $k\mathcal{P}$ and $\mathcal{P}$, i.e. the discrete logarithm problem.

In applications where only authentication is required, Schnorr's scheme [Sch89, Miy92] can be used which is as follows.

In Schnorr's scheme, apart from the working field $GF(q)$, the elliptic curve coefficients, base point and order, a hash function is also universally accepted. Let h denote the hash function which maps message $\mathcal{M} \in \mathbb{Z}$ to an integer less than the order of the curve.

$$h : GF(q) \otimes \mathcal{M} \longrightarrow \{0, \ldots, \#E(GF(q))\}$$

Suppose user A wants to send a message $\mathcal{M}$ to user B with his or her signature.

Signature Generation

1. Pick a random number k $(< \#E(GF(q)))$ and compute $\mathcal{Q} = k\mathcal{P} = (r_x, r_y)$.
2. Compute $e = h(r_x, \mathcal{M}) \in \{0, \ldots, \#E(GF(q))\}$.
3. Compute $y \equiv k - k_a e \ (mod\ l)$ and output the signature (e, y).

Signature verification

1. Compute $\mathcal{R} = y\mathcal{P} + e\mathcal{P}_a = (r_x, r_y)$ and check $e = h(r_x, \mathcal{M})$.

Hence we see that in Schnorr's signature generation requires one $k\mathcal{P}$ computation and signature verification requires two $k\mathcal{P}$ computations. Depending upon the requirements and constraints of an application, any of the two schemes can be used.

Chapter 4

Construction of Elliptic Curves

In this chapter, we discuss implementation aspects of theory discussed in Chapter 2 to construct elliptic curves with an order such that all the conditions discussed in Chapter 3 are met. We begin with an overview of a general procedure for construction of non-supersingular elliptic curves over $GF(q)$ (esp. over $GF(p)$ and $GF(2^n)$). Finally, we present algorithms for construction of elliptic curves over $GF(p)$ and $GF(2^n)$. We also discuss construction of supersingular elliptic curves over $GF(2^n)$ in the last section. First we review the main results of the previous chapter from the point of view of constructing elliptic curves with desired order (=#E($GF(q)$)).

4.1 Overview of Construction Procedure

From the theory discussed earlier, we know that every non-supersingular elliptic curve over a finite field has complex multiplication by an order $\mathcal{O}$ in a quadratic imaginary field $\mathcal{K}$ and the two norm equations will have a solution π in $\mathcal{O}$ which will be unique upto multiplication by a unit element in $\mathcal{O}$. In other words, for any non-supersingular elliptic curve E($GF(q)$) over $GF(q)$ with order #E($GF(q)$), there will exist a quadratic imaginary field $\mathcal{K}$ such that endomorphism ring of E($GF(q)$) will be isomorphic to an order in $\mathcal{K}$ and, moreover, there will be a $\pi \in \mathcal{O}$ such that

$$q = \mathrm{N}_{\mathbb{Q}}^{\mathcal{K}}(\pi) \tag{4.1}$$

and

$$\#\mathrm{E}(GF(q)) = \mathrm{N}_{\mathbb{Q}}^{\mathcal{K}}(1 - v_i\pi) \tag{4.2}$$

Here, $q = p^f$, $h_{\mathcal{K}} = h = fg$ and v_i is a unit in $\mathcal{O}$. The number of units in an imaginary quadratic field are 6,4 or 2 as the discriminant of imaginary quadratic field is $-3, -4$ or less than -4 [Hec93, Ros94, IR82, Cha88]. Now onward, we will write class number as h.

Equivalently, if we specify q and #E($GF(q)$) and try to search for a quadratic imaginary field $\mathcal{K}$ such that the two norm equations have a solution in $\mathcal{K}$, then we can be sure of

existence of an elliptic curve over $GF(q)$ such that it will have order as selected and its endomorphism ring will be isomorphic to an order in $\mathcal{K}$. But this information is not sufficient because for the construction of Weierstrass elliptic curve equation over finite field we need to know the j-invariant of the elliptic curve as well. For the computation of j-invariant, recall that the ring class field of quadratic imaginary field $\mathcal{H}_{\mathcal{O}}$ will be the splitting field (over $\mathcal{K}$) of j-invariants of isomorphic classes of elliptic curves having complex multiplication by $\mathcal{O}$ [BCh+66, Sil94, Shi71]. If we can compute the j-invariant for each isomorphism class, then $\mathcal{H}_{\mathcal{O}}$ will be the splitting field (over $\mathcal{K}$) of

$$f(x) = \prod_{i=1}^{h} (x - j(\mathrm{E}^{\sigma_i}))$$

This polynomial is referred as class equation (also as Weber's polynomial [Web02, LZ94, AM93]). Since, $j(\mathrm{E})$ is an algebraic integer, $f(x)$ will have coefficients in $\mathbb{Z}$. Hence, splitting field of this polynomial over $\mathbb{Q}$ will be field of modulii $\mathbb{Q}(j(\mathrm{E}))$.

If a rational prime p splits in $\mathcal{K}$ into prime ideals $\mathfrak{p}$ and $\mathfrak{p}'$ and, further, $\mathfrak{p}$ splits in $\mathcal{H}_{\mathcal{O}}$ in g prime ideals $\mathfrak{P}_i$ of relative degree f (with respect to $\mathfrak{p}$) [Ono90], then p will split in g prime ideals of relative degree f (with respect to p) in $\mathbb{Q}(j(\mathrm{E}))$. Hence, the polynomial $f(x)$ will split into g polynomials of degree f when reduced modulo p, i.e.

$$f(x) \equiv \varphi_1(x), \varphi_2(x), \ldots, \varphi_g(x) \mod p$$

where degree of $\varphi_i(x) = f$ for $1 \leq i \leq g$. Clearly, each $\varphi_i(x)$ will completely split in $GF(q)$ $(= \mathcal{O}_{\mathcal{H}_{\mathcal{O}}}/\mathfrak{P}_i = \mathcal{O}_{\mathbb{Q}(j(\mathrm{E}))}/p)$ [Ono90, IR82] and hence, we will get the j-invariant for the elliptic curve $\mathrm{E}(GF(q))$.

Now, we come to the computation of j-invariant for isomorphic classes of elliptic curves with complex multiplication by $\mathcal{O}$. As discussed earlier, there exists a one-to-one relationship between each ideal class of class group of $\mathcal{O}$ and isomorphism classes of elliptic curves. Hence, the problem of computation of j-invariant of an isomorphism class of elliptic curves is equivalent to computation of j-invariant of an ideal class of $\mathcal{CL}(\mathcal{O})$. Further, since each of the ideal class in $\mathcal{O}$ can be identified by a unique primitive reduced binary quadratic form with the discriminant same as that of $\mathcal{O}$ (see Appendix B), we need to look into the the relationship of $\mathcal{CL}(\mathcal{O})$ with set of primitive reduced binary quadratic forms [Ros94, IR82, Cha88, Hec93]. This relationship is very important for the other computational aspect as well, i.e. computation of class number of $\mathcal{O}$ [Ros94]. We will now concentrate on various computational aspects and will come back to problem of computing j-invariant later.

In the next section, we discuss the first step of curve construction procedure, i.e. solving norm equations, in detail.

4.2 Solving The Norm Equations

Solvability of the two norm equations 4.1 and 4.2 ensures existence of an elliptic curve over $GF(q)$ with order $\#\mathrm{E}(GF(q))$. The norm equation 4.1 specifies the finite field over which the constructed elliptic curve will be defined and Equation 4.2 specifies the order of the elliptic curve $\#\mathrm{E}(GF(q))$. Solution of the norm equations determines an imaginary quadratic field $\mathcal{K}$ such that an order in $\mathcal{K}$ will be isomorphic to endomorphism ring of elliptic curve with order $\#\mathrm{E}(GF(q))$.

Since our aim is to construct elliptic curves over finite fields with specified orders following the constraints discussed in Chapter 3, we would like to specify q for $GF(q)$ and the order of the curve $\#\mathrm{E}(GF(q))$ and search for a discriminant such that the two norm equations will have proper solution. In other words, we need to search a discriminant d such that there exists a $\pi = x + \sqrt{d}y$ $(2x, 2y \in \mathbb{Z})$ in an order $\mathcal{O}$ (of discriminant d) of an imaginary quadratic field $\mathcal{K}(\sqrt{\textit{square free part of } d})$ with *norm of* $\pi = x^2 + dy^2 = q$ and *norm of* $(1 - \pi) = (1 - x)^2 + dy^2 = m = \#\mathrm{E}(GF(q))$. This argument leads us to following theorem.

Theorem 4.1 *There exists $h(t^2 - 4p)$ isomorphism classes of non-supersingular elliptic curves over $GF(p)$ with order $\#E(GF(p)) = p + 1 - t$. Here $h(d)$ denotes the class number of discriminant d.*

Here it should be noticed that $dy^2 = (q + 1 - m)^2 - 4q$, and hence d will be a factor of right hand side of this equation. In fact, d can be taken as the square free part of the number on right hand side. In such a case, the conductor [Sil94, PS92, Coh78] of order $\mathcal{O}$ with discriminant d will be 1. Hence it will be maximal order $\mathcal{O}_\mathcal{K}$ in $\mathcal{K}(\sqrt{d})$ and corresponding ring class field will actually be Hilbert Class Field,i.e maximal unramified abelian extension of $\mathcal{K}(\sqrt{d})$ [PS92, Coh78](also see Appendix C). But here the problem is that the class number of this discriminant may be very large which in turn will mean that to obtain j-invariant a polynomial of a very large degree will have to be factored. One way to overcome this problem is that we first determine the discriminant as square free part $(q + 1 - m)^2 - 4q$ and then check if it has small class number (using algorithm pbqf_cln(d) given in Section 4.3). If the class number is small then select it otherwise try again with different q or m or both. In fact, if instead of giving both the parameters q and $\#\mathrm{E}(GF(q))$, if we give any one of them, i.e. either finite field $GF(q)$ or $\#\mathrm{E}(GF(q))$, then it is very easy to find a discriminant in a short time such that a π is obtained in $\mathcal{O}$ for the equation with the given parameter and the *norm of* $(1 - v_i\pi)$ is an integer satisfying all the requirements for the second parameter.

To justify this claim, let us recall the Hasse's theorem which says that order of an elliptic curve over $GF(q)$ is bounded by $q + 1 - 2\sqrt{q} \leq \#\mathrm{E}(GF(q)) \leq q + 1 + 2\sqrt{q}$. Hence, if q is large enough then there will be many choices for the $\#\mathrm{E}(GF(q))$ which will meet all the

required restrictions. In other words, probability of finding a discriminant will be very high. Likewise, for a given order, there will be many choices for q.

Hence, for construction of elliptic curves over prime field $GF(p)$, we will specify the order which satisfies all the restrictions for security and try to search for a discriminant such that second norm equation has a solution for which first norm equation gives a prime. This prime will define the working field. In case of curve construction over $GF(2^n)$, the field parameter is known. We will search for a solution of first norm equation such that second norm equation gives an integer satisfying all the restrictions for the order of elliptic curve as discussed in Chapter 3. In the sequel, we will see some constraints over the discriminants to be tested for solvability of norm equations which will help in formulating a procedure for efficient search of a proper discriminant. Now we discuss two cases , construction of elliptic over prime field $GF(p)$ and over $GF(2^n)$, separately in detail.

For curves over prime field $GF(p)$

For construction of curves over $GF(p)$, we specify the order in the form #E($GF(p)$)= $m = c * q$, where c is a small number (to say, less than 100) and q is a large prime such that $q-1$ has no factor less than an integer B. Here B is a lower bound on the minimum degree of extension for MOV attack to be applicable. For security purposes, the size of q should be at least 40 decimal digits and B should be at least 10. Now, for m to be norm in some quadratic field $\mathcal{K}$, each of prime factors of m must split in principal ideals in prime ideals in $\mathcal{K}$. Now, let us discuss the *Cornacchia's algorithm* [AM93] for solving the norm equation. This algorithm works for m which splits in principal ideals in $\mathcal{K}$. As discussed above, problem of solving norm equation is equivalent to finding $x, y \in \mathbb{Z}$ such that for a negative discriminant d,

$$4m = x^2 - dy^2.$$

Since discriminant d can either be 0 *or* 1 *mod* 4 [Ros94, PS92, Cha88], this equation can be modified in different cases as given below. If $d \equiv 0 \ mod\ 4$ then on putting $d = -4D$, we get

$$m = x'^2 + Dy'^2$$

If $d \equiv 1 \ mod\ 8$ then x and y will be even integers, hence putting $d = -D$, we get

$$m = x''^2 + Dy''^2$$

We shall see the last case $d \equiv 5 \ mod\ 8$ after the algorithm.

Procedure Cornacchia(D,m)
(** solution of $m = x^2 + Dy^2$ **)

1. Begin

2. Find x_0 such that $x_0^2 \equiv -D \ mod \ m$ and it also satisfies $m > x_0 > m/2$.

3. Develop m/x_0 as continued fraction

$$\begin{aligned} m &= q_0 x_0 + x_1, \\ x_0 &= q_1 x_1 + x_2 \\ &\cdots \\ x_r &= q_{r+1} x_{r+1} + x_{r+2} \end{aligned}$$

and stop when $x_r^2 < m \leq x_{r-1}^2$.

4. Set $x = x_r$ and $y = \sqrt{\frac{m - x^2}{D}}$

5. If y is not an integer, m is not representable as $x^2 + Dy^2$.

6. **end.**

In case of $d \equiv 5 \ mod \ 8$, same algorithm is used with x_0 as a solution of $x^2 + x + \sqrt{\frac{D+1}{4}} \ mod \ m$.

Hence, this algorithm finds the representation of $m = x^2 + Dy^2$ whenever one exists. Once a solution is obtained, i.e. $1 - v_i\pi$, we compute norm of $v_i\pi$ for all the units. If the norm value is prime number and $p^{2\phi(c)} \not\equiv 1 \ mod \ m$ as discussed in Chapter 3, then we select this discriminant as proper otherwise we go for another. From Hasse's Theorem, we can easily deduce that size of the field prime will be almost same as that of order $\#\mathrm{E}(GF(p))$. We will search the discriminant in increasing order of their class number because for obtaining the j-invariants a polynomial (class equation) of degree equal to class number will have to be factored (modulo p). Hence to reduce the computational complexity, we prefer discriminants with smaller class numbers. Within the set of discriminants having the same class number the search will be in increasing order of their magnitude. We will discuss this aspect in detail later in Section 4.6.

For curves over the field $GF(2^n)$

In this case, our objective is to construct elliptic curves over a given field $GF(2^n)$. We begin with solving Equation 4.1 for $q = 2^n$ ($f = n$) and then search for a discriminant (and hence π) such that Equation 4.2 gives an integer $m(=\#\mathrm{E}(GF(2^n)))$ which splits in c (a small integer, say less than 100) and a large prime q'. Moreover, the order m should be such that the prime factor $q' - 1$ is B-nonsmooth, i.e. has no factor less than integer B, and $q^{2\phi(c')} \not\equiv 1 \ mod \ m$ for c' equal to odd part of c. As discussed earlier, for security purposes, extension degree n of the field should be atleast 130. Any B greater than 10 will make MOV attack infeasible.

Now, we discuss the solvability of the first norm equation. As we mentioned earlier, there will exist a π in some quadratic field $\mathcal{K}(\sqrt{d})$ if and only if q is a norm of a principal ideal in $\mathcal{K}(\sqrt{d})$. Let's say 2 splits in $\mathcal{K}(\sqrt{d})$ into prime ideals $\mathfrak{p}$ and $\mathfrak{p}'$. That is,

$$(2)_{\mathcal{K}} = \mathfrak{p}\mathfrak{p}'$$

Here, if $\mathfrak{p}$ is not a principal ideal in $\mathcal{K}(\sqrt{d})$, then Cornacchia algorithm will fail for $m = 2$. But if the discriminant is such that its class number is n, then $\mathfrak{p}^n$ will definitely be principal as $\mathfrak{p}$ is an element of class group $\mathcal{CL}(\mathcal{O})$ which is cyclic group of order n.

Now we can think of many variations. As we have already discussed that $\mathcal{K}(j(\mathrm{E}))$ will be splitting field of a class equation $f(x)$, which is a polynomial of degree equal to class number of an order with discriminant same as that obtained from the norm equations. We will discuss the method for the computation of class equation later. This polynomial, when reduced mod 2, will split into g polynomials of degree $f(= h/g,\ h$ is the class number). Here, f will a minimum integer such that $\mathfrak{p}^f$ becomes a principal ideal in $\mathcal{K}(\sqrt{d})$ [PS92, Ono90, Hec93, IR82]. Now, consider the cases given below, with the following notations:

n = extension degree of working field $(GF(2^n))$ over $GF(2)$.

m =#E$(GF(2^n))$

$h = fg$ = class number of discriminant d for which norm equations are solvable and, 2 splits in $\mathcal{K}(\sqrt{d})$ into prime ideals $\mathfrak{p}$ and $\mathfrak{p}'$

1. *n is a composite number with h as a factor* : Let $n = ht$. Then we will look for a discriminant such that it has class number h and gives a proper π with $q = 2^n$ in the norm equations. Obviously, if 2 splits in $\mathfrak{p}$ and $\mathfrak{p}'$ in $\mathcal{K}(\sqrt{d})$ then 2^n will split in principal ideals in $\mathcal{K}(\sqrt{d})$ as

$$2^n = (2^h)^t = (\mathfrak{p}^h)^t(\mathfrak{p}'^h)^t$$

   Since $\mathfrak{p}^h$ is principal ideal, $\mathfrak{p}^{ht=n}$ will also be principal. Now, the polynomial $f(x)$ will be of degree h and, when reduced modulo 2, will either be an irreducible polynomial of degree h or will split into g polynomials of degree f. Since, the j-invariant for the curve will be a root of $f(x)$, in this case, we will require to factor a polynomial of small degree over $GF(2^n)$. Hence, in this case, we can define the working field $GF(2^n)$ (i.e. a primitive irreducible polynomial of degree n over $GF(2)$) and can factor $f(x)$ for j-invariant. The factorization will not be difficult computationally as the degree of $f(x)$ will be small.

2. *n is equal to h* : Here, again there will be two cases

   (a) *n is prime* : In such a case, if 2 splits in prime ideals in $\mathcal{K}(\sqrt{d})$ then 2^n will split in principal ideals $\mathfrak{p}^n$ and $\mathfrak{p}'^n$. Hence, the class equation $f(x)$ *mod* 2 will be an

is a 2×2 matrix $M(f)$ associated with each binary quadratic form $f(x, y)$ such that

$$f(x, y) = XM(f)X^T$$

where $X = [x, y]$ and $M(f) = \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix}$.
If D is the determinant of the $M(f)$ then $d = -4D$.
The two binary quadratic forms with matrices A and B are said to be equivalent if there exists a unimodular 2×2 matrix T such that $B = TAT^{-1}$. If the determinant of T is equal to -1 then the two forms are called improperly equivalent. With this equivalence operation, the set of all quadratic forms with same discriminant can be divided into finite equivalence classes and each equivalence class will be represented by a primitive reduced from, i.e. a form (a,b,c) with gcd(a,b,c) equal to 1 [Dav80, IR82, Ros94, AM93].Moreover, it is also possible to define a binary composition law on the equivalence classes to give a group structure to the set of equivalence classes. [SHL84]

Theorem 4.2 given below, sets up an isomorphism between the set of primitive reduced binary quadratic forms of discriminant d and a class group $\mathcal{CL}(\mathcal{O})$ of an order $\mathcal{O}$ with discriminant d of an imaginary quadratic field [AM93, SHL84].

Theorem 4.2 *For an order $\mathcal{O}$ in a quadratic field (real or imaginary) with discriminant d, the map sending a fractional ideal $[1, \tau]$ in $\mathcal{CL}(\mathcal{O})$ to a primitive reduced binary quadratic form (a,b,c) of discriminat d such that $\tau = (\frac{-b+\sqrt{d}}{2a})$ induces an isomorphism between $\mathcal{CL}(\mathcal{O})$ and group of equivalence classes of quadratic forms $\mathcal{C}(d)$ with discriminant d, i.e. tha map*

$$\begin{aligned} \phi : \mathcal{CL}(\mathcal{O}) &\longrightarrow \mathcal{C}(d) \\ [\tau, 1] &\longmapsto [x, y] = [(-b + \sqrt{d})/2a, 1] \end{aligned}$$

is an isomorphism. Moreover the composition law of equivalence classes of quadratic forms is equivalent to multiplication of fractional ideals in $\mathcal{CL}(\mathcal{O})$.

Since we are concerned with imaginary quadratic fields only, now onward we will concentrate over quadratic forms with negative discriminant. The following three rules determine the primitive reduced form of each equivalence class of set of quadratic forms with discriminant d ($d < 0$) [AM93, Ros94].

1. $|b| \leq a \leq c$
2. $c > 0$ and $0 < a \leq \sqrt{|d|/3}$
3. $(a, b, c) \sim (a, -b, c)$ iff either $a = c$ or $a = |b|$

Obviously, the number of primitive reduced binary quadratic forms will be equal to class number of $\mathcal{CL}(\mathcal{O})$. Hence, this method avoids the use of Dirichlet formula [Dav80, Hec93, Ros94, IR82] for the computation of class number of an order $\mathcal{O}$ in any imaginary quadratic fields (for real quadratic field as well, but with different rules) which is computationally very expensive.We now present an algorithm for finding all the primitive reduced binary quadratic forms and class number for a given discriminant d.

Procedure pbqf_cln(d)

(** pbqf_cln stands for Primitive Binary Quadratic Forms and CLass Number **)

1. **Begin**
2. Set $class\_no := 0$ and $D := |d|$.
3. Set $r = \lfloor\sqrt{D/3}\rfloor$ and $b = D \bmod 2$.
4. while $(b \leq r)$
   (a) Set $a := 1$ or 0 and $b = 0$ or 1 respectively.
   (b) while $(a \leq \lfloor\sqrt{(b^2 + D)/4}\rfloor)$
      i. if $((m = 0 \bmod a)\ and\ (b \leq a))$
         A. Set $c := m/a$.
         B. if $(gcd(a, b, c) = 1\ and\ (c = a\ or\ b = a\ or\ b = 0))$
            Set $class\_no := class\_no + 1$, Store primitive form (a, b, c).
         C. if $(gcd(a, b, c) = 1\ and\ (c \neq a\ and\ b \neq a\ and\ b \neq 0))$
            else
            Set $class\_no := class\_no + 2$, Store primitive forms $(a, \pm b, c)$.
      ii. Set $a := a + 1$.
   (c) Set $b := b + 2$.
5. **end.**

Using this algorithm we can obtain all the primitive quadratic forms which will represent an ideal class in $\mathcal{CL}(\mathcal{O})$. Hence a large part of the ideal theory can be translated into the language of the theory of quadratic forms. We will see in the next section that all the computation in $\mathcal{O}$ will be carried out in terms of corresponding primitive binary quadratic forms. To a large extent the theory of quadratic forms is concerned with the problem of which numbers can be represented by the form $f(x, y)$ if x, y run through all the pairs of integers in $\mathbb{Z}$. This is equivalent to the problem of which numbers appear as norm of (principal) integral ideals in a given ideal class discussed in Section 4.2.

4.4 Computation of j-invariants and Class Equation

In Section 4.2, we obtained a discriminant d such that corresponding order $\mathcal{O}$ will be isomorphic to an endomorphism ring of elliptic curve ($\mathrm{E}(GF(p))$ or $\mathrm{E}(GF(2^n))$) with a known order ($\#\mathrm{E}(GF(p))$ or $\#\mathrm{E}(GF(2^n))$). In this section, we begin with computation of j-invariants for various isomorphism classes of elliptic curves (over $\mathbb{C}$) having $\mathcal{O}$ as endomorphism ring.

As discussed in Section 2.1, each isomorphism class of elliptic curves is identified by a homothety class of a lattice $\Lambda = \mathbb{Z}\tau + \mathbb{Z}$ and that the corresponding j-invariant is unique. Since each homothety class of lattice can be represented by a fractional ideal in an order $\mathcal{O}$ of an imaginary quadratic field, the j-invariant for an isomorphism class of elliptic curve will be same as that of an ideal class in $\mathcal{CL}(\mathcal{O})$.

Now, to obtain an expression for computation of $j(\tau)$ for an ideal class, we define Dedekind's η function and Weber's f, f_1, f_2 functions [Web02, Sil94, AM93, LZ94, PS92].

Definition 4.1 *The Dedekind η-function $\eta(\tau)$, for a τ lying in upper half of complex plane, is defined by the product*

$$\eta(\tau) = q^{1/24}\prod_{n\geq 1}(1-q^n), \qquad for \quad q = e^{2\pi i\tau}.$$

This function is a modular form of weight $1/2$ and can be expanded as

$$\eta(\tau) = q^{1/24}(1+\sum_{i\geq 1}(-1)^n(q^{n(3n-1)/2}+q^{n(3n+1)/2})) \tag{4.3}$$

If we let ζ_n stand for $\exp(2i\pi/n)$, then the Weber's functions are defined as

$$f(\tau) = \zeta_{48}^{-1}\frac{\eta((\tau+1)/2)}{\eta(\tau)}, \tag{4.4}$$

$$f_1(\tau) = \frac{\eta(z/2)}{\eta(\tau)}, \tag{4.5}$$

$$f_2(\tau) = \sqrt{2}\frac{\eta(2\tau)}{\eta(\tau)}. \tag{4.6}$$

Further, we define

$$\gamma_2(\tau) = \frac{f^{24}(\tau)-16}{f^8(\tau)} = \frac{f_1^{24}(\tau)-16}{f_1^8(\tau)} = \frac{f_2^{24}(\tau)-16}{f_2^8(\tau)}, \tag{4.7}$$

$$\gamma_3(\tau) = \frac{(f_2^{24}(\tau)+8)(f_1^8(\tau)-f_2^8(\tau))}{f^8(\tau)}. \tag{4.8}$$

Using these functions, the modular invariant $j(\tau)$ for an ideal class of $\mathcal{O}$ can be computed as

$$j(\tau) = \gamma_2^3(\tau) = \gamma_3^2(\tau) + 1728. \tag{4.9}$$

For further details regarding these functions, please see [Sil94, Web02, LZ94, PS92]. Using these functions, the j-invariants for each ideal class of $\mathcal{CL}(\mathcal{O})$ can be computed. Obviously, there will be $h(= \#\mathcal{CL}(\mathcal{O}))$ such j-invariants as there are h distinct ideal classes in $\mathcal{CL}(\mathcal{O})$.

Above discussion does tell us a way for the computation of j-invariants but is not sufficient from the point of actual computation over a finite precision computing machine as computation of j-invariants involves floating point arithmetic and sum of a infinite series. For computations over a finite precision computation machine, we need to expound over its implementation aspects in detail. In Section 4.3, we established an isomorphism between $\mathcal{CL}(\mathcal{O})$ and group of binary quadratic forms with the same discriminant $\mathcal{C}(d)$. For a fractional ideal $[\tau, 1]$ in an ideal class of $\mathcal{CL}(\mathcal{O})$, the τ can be expressed in terms of coefficients of corresponding primitive quadratic form (a, b, c) as

$$\tau = \frac{-b + \sqrt{b^2 - 4ac}}{2a} = \frac{-b + \sqrt{d}}{2a}$$

But to determine the j-invariant, we require to compute the $\eta(\tau)$ function which is an infinite series. Moreover, since the class equation $f(x)$ is an irreducible polynomial over $\mathbb{Q}$, the j-invariants will be complex number with real and imaginary part in $\mathbb{R}$. Hence, for computational purposes we must define the floating point precision so that after rounding off the coefficients of $f(x)$ are actual, i.e. same as those which will be obtained with infinite precision arithmetic [AM93]. By floating point precision, we mean the number of significant decimal digits to be preserved in floating point arithmetic. Of course, from the computational point of view, it will be desirable to have minimum floating point precision. Following example illustrates clearly the meaning of finite floating point precision.

Example

Let $x = 234.32873456667$ and $y = 8768.785478947$.

Then, $x * y = \underline{2054778}.404968241\ldots$

Now for $x * y$ to be correct to nearest integer value, it can be computed with less number of significant digits after decimal places in x and y. If we compute the product with x and y considered till 5th place after decimal(because one of them has 4 digits in integer part), the product will remain same to nearest integer value as

$234.\underline{32873} * 8768.\underline{78547} = \underline{2054778}.3628275531.$

Whereas if x and y are taken till third decimal place only then product will not be correct to nearest integer value as

$234.\underline{328} * 8768.\underline{785} = \underline{2054771}.851\ldots$.

Likewise, instead of two numbers if we have n integers with at most t digits in their integer portion, we need to preserve each number till ,at least, $(t + 1) * (n - 1)$th decimal place for their product to be accurate to nearest integer value. In fact, a tighter bound can be obtained if we define the finite precision to be equal to sum of digits in integer part of all

the multiplicands. It is better to have floating point precision greater than this to avoid the possibility of error in rounding off.

Let us first estimate the approximate size of magnitude of j-invariants in decimal digits. The following expression gives the q-expansion of $j(\tau)(= j(\mathrm{E}_{\Lambda_\tau}))$ [Sil94, AM93, Apo76]

$$j(\tau) = \frac{1}{q} + \sum_{n\geq 0} c(n)q^n,$$

where $c(n) \in \mathbb{Z}$ and can be expressed in terms of Ramanujan's $\tau(n)$ [Sil94] function (Silv2). Here, we give first few values of $c(n)$.

$c(0) = 744$, $c(1) = 196884$,

$c(2) = 21493760$, $c(3) = 864299970$

Now, since $q = e^{2\pi i\tau}$, on substituting $\tau = \frac{-b+i\sqrt{|d|}}{2a}$ we get

$$|q^{-1}| = e^{\frac{\pi\sqrt{|d|}}{a}}$$

where, $d = b^2 - 4ac$. The main term in $j(\tau)$ is $1/q$, hence we get an approximation for magnitude of $j(\tau)$ as

$$|j(\tau)| = |q^{-1}| = e^{-\frac{\pi\sqrt{|d|}}{a}}.$$

The approximate number of decimal digits in $j(\tau)$ is

$$\log|j(\tau)| \approx \frac{\pi\sqrt{|d|}}{a\ln 10}.$$

Now, since the constant term of class equation is product of all the j-invariants, a floating point precision which is appropriate for this term, will suffice for all others as they involve product of fewer number of j-invariants. Hence, appropriate precision for floating point arithmetic can be given by

$$Prec(d) = \sum_{i=1}^{h} \frac{\pi\sqrt{-d}}{a_i log 10} + f(h) + C_h.$$

Here, $f(h) + C_h$ has been added to compensate for the error in computation of precision. The function $f(h)$ is an increasing function of class number and C_h is some constant. In our implementations, we have used $f(h) = h/4$ and $C_h = 10$. Hence, we need to preserve any number to $Prec(d)$ significant decimal digits in computations. Since, $\eta(\tau)$ can be expressed as a power series of q and magnitude of q^n converges to zero with increasing value of n, we need to consider only first N terms for the computation of $\eta(\tau)$ such that terms of higher powers become insignificant for above defined precision. To compute N, we define $\eta_N(\tau)$ as the sum of first N terms of $\eta(\tau)$.

$$\eta_N(\tau) = q^{1/24}(1 + \sum_{n=1}^{N}(-1)^n(q^{n(3n-1)/2} + q^{n(3n+1)/2})),$$

It can easily be shown that

$$|\eta(\tau) - \eta_N(\tau)| \leq 6c^{\frac{\pi\sqrt{-d}(1-36N^2)}{24a}}.$$

Taking log of both sides, we get

$$\log|\eta(\tau) - \eta_N(\tau)| \ln 10 \leq \log 6 + \frac{\pi\sqrt{-d}(1-36N^2)}{24a}.$$

Now, according to precision function defined above

$$\begin{aligned} 10^{-Prec(d)} &\geq |\eta(\tau) - \eta_N(\tau)| \\ \Rightarrow \quad \log|\eta(\tau) - \eta_N(\tau)| &\leq -Prec(d) \end{aligned}$$

Combining the above two equation we get,

$$\begin{aligned} -Prec(d)\ln 10 &\geq \log 6 - \frac{3\pi\sqrt{-d}N^2}{2a} + \frac{\pi\sqrt{-d}}{24a} \\ \Rightarrow \quad N &\geq \sqrt{\frac{2}{3}\frac{(\log 6 + \ln 10 Prec(d))}{\pi\sqrt{-d}} + \frac{1}{24}} \end{aligned}$$

Hence, for the computation of $\eta(\tau)$ we need to consider minimum number of terms as given by the above expression. Similarly, if we want to compute $\eta(k\tau)$, then

$$N \geq \sqrt{\frac{2}{3}\frac{(\log 6 + \ln 10 Prec(d))}{\pi k\sqrt{-d}} + \frac{1}{24}}.$$

Here, for each value of τ (and hence for isomorphism class of elliptic curves), N is to be computed. Once N is known, j-invariants can easily be computed using Weber's function. In fact, we need not compute j-invariants for all the isomorphism classes as the binary primitive forms (a, b, c) and $(a, -b, c)$, where $a \neq |b|$ and $a \neq c$, will represent two different ideal classes and in such a case j-invariant for one class can be computed from the other.

$$j(a, -b, c) = \overline{j(a, b, c)}$$

Here, bar indicates the complex conjugation. Once all the j-invariants are computed, the class equation can easily be computed. If the used floating point precision is appropriate then the coefficient will be real and very close to integer value. The correct integer coefficients can be obtained by rounding them off to nearest integer value.

For the construction of curves over $GF(2^n)$, we may require to select the discriminant which has class number equal to degree of extension n of $GF(2^n)$ over $GF(2)$. Since for security reasons, n should be at least 140, the precision will be very high as the summation is carried over all the ideas classes. Even in case of constructing curve over $GF(p)$, the

discriminants may have a large class number. From the computational point of view, it is desirable to have floating point precision as minimum as possible. If the discriminants with some particular class number are of concern, then the discriminants with minimum absolute value should be selected as the precision is directly proportional to the square root of it. This justifies the reason for searching the discriminants in ascending order of their absolute value. Though, a also appears in the expression for computation of precision, we can not put any restriction over it as nothing can be deduced about the values of a for ideal classes prior to search of the discriminants.

For (further) reduction in precision required for a given discriminant, *Weber's class invariant* [Web02, BCh+66, AM93, LZ94] plays an important role. Weber calls any function $u_?(\tau)$ a class invariant if its splitting field is $\mathcal{H}_\mathcal{O}$. In other words,

$$f(x) = \prod_{i=1}^{h}(x - u_?(\tau)^{\sigma_i})$$

will have integer coefficients and $\mathcal{H}_\mathcal{O}$ will be splitting field of $f(x)$ (like for j-invariants). In fact, the class invariant is a generalization of the concept of the j-invariant. For computational purposes, we are interested in class invariants $u_?(\tau)$ which require low precision for floating point arithmetic and they are such that the j-invariants can be expressed in terms of them.

Weber obtained the following class invariants with some restrictions over the discriminants.

- $u_0(\tau) = j(\tau)$, if there is no restriction over the discriminants. Hence, the precision will be same as obtained above. Now, if the root of class equation $f(x)$ mod q is x_0, then the j-invariant over $GF(q)$ will be x_0.

- If $d \not\equiv 0 \ mod\ 2$, then $u_1(\tau) = \sqrt{d}\zeta_3^{(a-c+a^2c)b}\gamma_2(\tau)$ is a class invariant. Using equations 4.7 and 4.9, we can express $j(\tau)$ in terms of $u_1(\tau)$ as

  $$j(\tau) = u_1^2(\tau)/d + 1728.$$

  Now, the new precision $Prec'(d)$ can be expressed in terms of, $Prec(d)$ as given below

  $$Prec'(d) = \frac{Prec(d) + h\log|d|}{2}.$$

  In this case, required precision is half of what was computed for j-invariants. Further, the j-invariant over $GF(q)$ will be $\frac{x_0^2}{d} + 1728$.

- If $d \not\equiv \ mod\ 3$, then $u_2(\tau) = (-1)^{(a+c+ac)(b-1)/2}\gamma_3(\tau)$ and the new precision value is

  $$Prec'(d) = Prec(d)/3.$$

  The j-invariant over $GF(q)$ will be x_0^3.

For $d \equiv 1 \ mod\ 8$ and $d \not\equiv 0 \ mod\ 3$, Yui and Zagier obtained another class invariant $u_3(\tau) = f^*(\tau)$, where

$$f^*(\tau) = \begin{cases} \zeta_{48}^{(a-c-ac^2)b} f(\tau) & \text{if } 2|a \text{ and } 2|c, \\ (-1)^{\frac{d-1}{8}} \zeta_{48}^{(a-c-ac^2)b} f_1(\tau) & \text{if } 2|a \text{ and } 2 \not| c, \\ (-1)^{\frac{d-1}{8}} \zeta_{48}^{(a-c+ca^2)b} f_2(\tau) & \text{if } 2 \not| a \text{ and } 2|c. \end{cases}$$

Since $j(\tau) = \gamma_2^3(\tau) = (\frac{f^{24}(\tau)-16}{f^8(\tau)})^3 = (\frac{f_1^{24}(\tau)+16}{f_1^8(\tau)})^3 = (\frac{f_2^{24}(\tau)+16}{f_2^8(\tau)})^3$,
the new precision function is

$$Prec'(d) \approx Prec(d)/48.$$

and j-invariant over $GF(q)$ is $\frac{(x_0^{24}-16)^3}{x_0^{24}}$.

This class invariant is very useful for constructing elliptic curves over $GF(2^n)$ as it reduces the required precision by a factor of 48. Moreover, 2 will split completely with the discriminant for this class invariant because $d \equiv 1 \ mod\ 8$.

So far, we have seen how to compute the class equation. Now, it remains to obtain the elliptic curve equation over $GF(q)$ which we discuss in the next section.

4.5 Computing the Curve Equation over $GF(q)$

Once we have the class equation, which is a polynomial with integer coefficients, we can obtain the j-invariants for the isomorphism classes of elliptic curve over finite field $GF(q)$ by factoring it over $GF(q)$.

If the constructed curve is defined over prime field $GF(p)$, i.e. if $q = p$ in the first norm equation, then the class equation $f(x)$ will completely split modulo p. Let x_i, $1 \leq i \leq h$, be the roots of this polynomial. Now, the corresponding $j_i(GF(q))$ will be as given below for different class invariants used in computing class equation $f(x)$.

1. $j_i = x_i$ for class invariant $u_0(\tau)$.

2. $j_i = x_i^2/d + 1728$ for class invariant $u_1(\tau)$.

3. $j_i = x_i^3$ for class invariant $u_2(\tau)$.

4. $j_i = (x_i^{24} - 16)^3/x_i^{24}$ for class invariant $u_3(\tau)$.

For curves over $GF(2^n)$, we select $u_3(\tau)$ as the class invariant. In this case, if we go for the options in which the class equation is an irreducible polynomial of degree n then this polynomial can be used for defining the field $GF(2^n)$. Hence, there will be no need to factor this polynomial for the computation of j-invariant, which is very difficult for large values of

n. If α is assumed to be its root in $GF(2^n)$ [LN94, Men93a, McE87] (i.e. $f(\alpha) = 0$), then all $\beta \in GF(2^n)$ can be written as

$$\beta = \sum_{i=0}^{n} \beta_i \alpha^i, \qquad \alpha_i \in GF(2)$$

Now, j_0 invariant can be computed as

$$\begin{aligned} j_0(GF(2^n)) &= (\alpha^{24} - 16)^3/\alpha^{24} \ mod \ f(\alpha) \\ &= \alpha^{48} \ mod \ f(\alpha). \end{aligned}$$

Since α^{2^i}, $1 \leq i \leq (n-1)$, will also be roots of class equation, other j_i's can be computed by raising α^{2^i} to power 48. In case, we go for the other option in which class equation needs to be factored over $GF(2^n)$, the same idea will still be applicable. That is, j-invariants will be obtained by raising the roots of the class equation to power 48.

Now recall that when we solve one norm equation, then there will be more than one solution, as all the associates[1] of any π will give the same norm value. Whereas, when the same π is put in the second norm equation, various values (equal to number of units in $\mathcal{O}$) are obtained for the second parameter. It means that over a given finite field there will be more than one choice for the order of the curve such that the endomorphism ring of all the curves over that field, having order same as any of those choices, will be same. Moreover, there will be h choices of j-invariants for a given discriminants, irrespective of what choice was taken as the order. Because of this, for each j-invariant there will be more than one isomorphism class of the elliptic curve and the number of these classes will be equal to number of units in the corresponding order $\mathcal{O}$. We know that for discriminants less than -4, number of units in order $\mathcal{O}$ are 2, hence for each j-invariant there will be two choices of curves. We call the second curve to be the *twist* of first [LZ94, Sil85]. The next theorem gives us the parameters of a curve and its twist [Sil85].

Theorem 4.3 *1. Let the prime $p > 3$ for $GF(p)$ and j-invariant $j_i \in GF(p)$ be given. Then corresponding elliptic curve over $GF(p)$, E and its twist $\tilde{E}$ are*

$$\begin{aligned} E : y^2 &= x^3 + ax + b \\ \tilde{E} : y^2 &= x^3 + \tilde{a}x + \tilde{b} \end{aligned}$$

where

$$\begin{aligned} a = \frac{3j_i}{1728 - j_i} \quad and \quad b = \frac{2j_i}{1728 - j_i} \qquad & if \ j_i \neq .0, 1728, \\ a \in GF^*(p) \quad and \quad b = 0 \qquad & if \ j_i = 1728, \\ a = 0 \quad and \quad b = \in GF^*(p) \qquad & if \ j_i = 0, \\ \tilde{a} = ac^2 \quad and \quad \tilde{b} = bc^3 \qquad & for \ any \ non\text{-}square \ c \ in \ GF(p) \end{aligned}$$

[1] $v_i\pi$, where v_i are units in $\mathcal{O}$

and $\#E(GF(p))+\#\tilde{E}(GF(p))= 2p+2$.

2. *For* $j_i \in GF(2^n)$, *the elliptic curve* E *and its twist* $\tilde{E}$ *are*

$$\begin{aligned} E: y^2 + xy &= x^3 + ax^2 + j_i^{-1}, \\ \tilde{E}: y^2 &= x^3 + \tilde{a}x + j_i^{-1} \end{aligned}$$

where $a, \tilde{a} \in GF(2^n)$ *such that* $Tr(a) + Tr(\tilde{a}) = 1$ *and following equalities hold*

$$\begin{aligned} \#E(GF(2^n)) + \#\tilde{E}(GF(2^n)) &= 2.2^n + 2, \\ \#E(GF(2^n)) \equiv 2Tr(a) \ mod\ 4 \quad &and \quad \#\tilde{E}(GF(2^n)) \equiv 2Tr(\tilde{a}) \ mod\ 4. \end{aligned}$$

Using this theorem, we can compute the coefficients of an elliptic curve equation and its twist. In case of curves over $GF(2^n)$, we first check whether the order of the curve $\#\mathrm{E}(GF(2^n))$ is congruent to 2 or 0 modulo 4. If $\#\mathrm{E}(GF(2^n)) \equiv 2 \ mod\ 4$ we select a such that its trace is 1, otherwise we take it as 0. The order of its twist $\tilde{\mathrm{E}}(GF(2^n))$ will be $2.2^n+2-\#\mathrm{E}(GF(2^n))$ and $\tilde{a}$ will an element of trace 0 or 1 as trace of a is 1 or 0 respectively.

For curves over $GF(p)$, we first determine a and b and then randomly find a point $\mathcal{P}$ on this curve. If $\#\mathrm{E}(GF(p))\mathcal{P}$ is point at infinity then it is the correct curve equation otherwise $\tilde{\mathrm{E}}(GF(p))$ will be the correct curve equation. The coefficients of $\tilde{\mathrm{E}}(GF(p))$ can be computed by finding any non-square c in $GF(p)$.

Hence, using this method we obtain $2h$ curves simultaneously having complex multiplication with an order in an imaginary quadratic field. In the next section, we explain the method for efficient search of discriminants for solving norm equations.

4.6 Dictionary of Discriminants and Class Numbers

The search of a discriminant for solving the norm equation is very important as we can be sure of getting a curve only if a proper discriminant exists. Since, for a (imaginary and real) quadratic field, discriminants of orders can either be congruent to 1 or 0 modulo 4, only such negative integers should be tested for solving the norm equations. Since, as discussed in earlier sections, in some cases discriminants following certain conditions only will be of interest, let us first summarize all of those conditions for constructing curves over $GF(p)$ (case a) and $GF(2^n)$ (case b);

1. (a & b) The discriminant $d(< 0)$ should either be 0 or $1 \equiv$ mod 4.

2. (a) Discriminants with small class number are preferred because the precision required will be less. Moreover, the computation of j-invariant (or class invariants) over $GF(p)$ will be easier as the degree of class equation will be small.

(b) Discriminants with some particular class numbers, as discussed in Section 4.2, are of interest.

3. (a & b) Discriminants with small absolute value are preferred as precision required for floating point arithmetic is directly proportional to $\sqrt{|d|}$.

4. (a) If, for the computation of the class equation, Weber or Yui-Zagier's class invariants are to be used then d should *either* be an odd integer ($u_1(\tau) = \sqrt{d}\zeta_3^{(a-c+a^2c)b}\gamma_2(\tau)$) *or* coprime to 3 ($u_2(\tau) = (-1)^{(a+c+ac)(b-1)/2}\gamma_3(\tau)$) *or* $1 \equiv mod\ 8$ and coprime to 3 ($u_3(\tau) = f^*(\tau)$). If the j-invariants are used for class equation then there will be no restriction.

   (b) Discriminants should be congruent to 1 modulo 8 otherwise 2 will not split in two distinct prime ideals in $\mathcal{K}(\sqrt{d})$. Since we are interested in large fields, Yui-Zagier's class invariant should be used for low value of precision. Hence, d should also be coprime to 3.

For efficient search of the discriminant we build a dictionary which contains the negative discriminants from -3 to some large value (say -100000), which follow the condition (1). The dictionary also contains class numbers for each discriminant , which is computed using algorithm **pbqf_cln(d)** described in Section 4.3. The dictionary is sorted in increasing order of $|d|$. For construction of curve over $GF(p)$, we begin with extracting all the discriminants of class number 1 and test for solvability using *Cornacchia's algorithm.* If proper discriminant is not found, then all the discriminant of class number 2 are extracted out from the dictionary for solving norm equations and so on. As of now, it is not known that how many discriminants are there for a given class number except for class number 1 for which there are 13 negative discriminants. Hence, an increase in size of dictionary will increase the probability of finding the proper discriminant quickly, though in our implementation we have found -1000000 as sufficient. In case of curve construction over $GF(2^n)$, we need to extract the discriminants of a particular class number only which is either equal to, or divides, or is divisible by the extension degree n of $GF(2^n)$ over $GF(2)$ as discussed in Section 4.2.

In the next section, we conclude the discussion with algorithms for construction of non-supersingular elliptic curves over $GF(p)$ and $GF(2^n)$.

4.7 Curve Construction Algorithms

In this section, we present the final algorithms for constructing non-supersingular elliptic curves over $GF(p)$ and $GF(2^n)$ based on what we have discussed so far. For construction of curves a dictionary containing negative discriminants and corresponding class numbers, as

discussed in Section 4.6, is used for efficient search of discriminants.

Curve construction over $GF(p)$

In this case, instead of specifying the order of the curve we specify the upper limit of small factor and size of the prime factor of the order.

Input

- n = Number of decimal digits in prime factor of #E($GF(p)$),
- B = Lower bound on minimal extension degree for MOV attack,
- C = Upper limit of small factor of order.

Output

- the prime p for working field $GF(p)$,
- a and b, coefficients of the constructed elliptic curve E:$y^2 = x^3 + ax + b$,
- #E($GF(p)$), the order of the constructed curve. The order will have a large prime factor q such that $(q-1)$ will be B-nonsmooth,
- j-invariant of the constructed curve.

Procedure ECFP(n,B,C)

1. Begin
2. Find randomly an n digit prime q such that $q-1$ is B-nonsmooth.
3. Set i:=1.
4. Extract all discriminants of class number i from dictionary. If dictionary is exhausted then augment the size of the dictionary otherwise Set j:=1 and Continue.
5. If all the discriminant of class number i are not exhausted then assign the j^{th} of the extracted discriminants to d and Set c:=1, otherwise increment i and Go to 4.
6. if $c \leq C$ then Continue, otherwise increment j and Go to 5.
7. Call Cornacchia$(c * q, d)$.
8. If Cornacchia() succeeds assign the solution to π otherwise increment c and Go to 6.
9. Check whether norm of $(1 - v_i\pi) = p$ is prime for any of units v_i in $\mathcal{O}$. If p is prime and $p^{2\varphi(c)} \not\equiv 1 \ mod(c * q)$ then continue, otherwise increment c and Go to 6.

10. Compute all the primitive binary form using pbqf_cln(d).

11. Compute the precision $Prec(d)$ as given in Section 4.4.

12. Choose a proper class invariant $u_?$ depending upon the discriminant and accordingly obtain new reduced precision $Prec'(d)$.

13. Set $f(x) = 1$;

14. for k=1 to i

    (a) Obtain N for computation of η_N corresponding to k^{th} binary quadratic forms.

    (b) Compute $\eta(\tau_k)$ till first n terms with $Prec'(d)$ precision.

    (c) Compute the chosen class invariant $u_?(\tau_k)$.

    (d) Compute $f(x) = f(x) * (x - u_?(\tau_k))$ with above defined precision.

15. Round off the coefficients of f(x) to nearest integer.

16. Factor f(x) modulo p to obtain roots $x_t, 1 \leq t \leq i$.

17. Pick any x_t and compute j_t, the j-invariant, according to chosen class invariant.

18. Compute a and b as given in Section 4.5.

19. Find out a point $\mathcal{P}= (x, y)$ on E:$y^2 = x^3 + ax + b$ by solving this equation for y for any random value of x.

20. Compute $(c * q)\mathcal{P}$. If $(c * q)\mathcal{P}$ is point at infinity, then a and b are correct coefficients, otherwise find a non-square in $GF(p)$ and compute $\tilde{a}$ and $\tilde{b}$ which will be the correct coefficients.

21. End

Curve construction over $GF(2^n)$

In this case, we need to search discriminants of some particular class number as given in Section 4.2. Here, we briefly discuss different cases following the notation as used in Section 4.2.

1. $n = ht$: In this case, we need to consider discriminant of class number h only and class equation will be a polynomial of degree h (not necessarily irreducible over $GF(2)$). For obtaining j-invariant the class equation will have to be factored.

2. $n = h$:

(a) n is prime: Here, the discriminant of class number n need to be searched and class equation will definitely be an irreducible polynomial over $GF(2)$. Hence, the class equation can be taken as modulo polynomial for defining $GF(2^n)$.

(b) n is composite: If n is composite then class equation may be reducible over $GF(2)$ and hence j-invariants will be obtained by finding roots of class equation in $GF(2^n)$; otherwise class equation can be taken as modulo polynomial for defining $GF(2^n)$.

3. $h = nt$: In this case, class equation will be a polynomial of degree h and will be reducible over $GF(2)$. If its factors are irreducible polynomials of degree n over $GF(2)$(which will surely happen if n is prime) then any of these factors can used for defining $GF(2^n)$ to avoid the factorization for j-invariant; otherwise the class equation will have to be factored over $GF(2^n)$.

In view of these points, the algorithm will be as follows.

Input

- n = The extension degree of $GF(2^n)$ over $GF(2)$,
- h = The class number of the discriminants to be searched,
- B = Lower bound on minimal extension degree for MOV attack,
- C = Upper limit of small factor of order.
- H = The modulo polynomial for $GF(2^n)$, if needed.

Output

- the modulo polynomial H for $GF(2^n)$ (if not given in input),
- a and b, coefficients of the constructed elliptic curve E($GF(2^n)$):$y^2 + xy = x^3 + ax^2 + b$,
- #E($GF(2^n)$), the order of the constructed curve. The order will have a large prime factor q such that $(q-1)$ will be B-nonsmooth,
- j-invariant of the constructed curve.

Procedure ECGF2(n,h,B,C,H)
(** H is optional **)

1. **Begin**

2. Extract all the discriminants of class number h from dictionary. If there is no discriminant then augment the size of the dictionary otherwise Set j:=1 and Continue.

3. If all the discriminant of class number h are not exhausted then assign the j^{th} of the extracted discriminants to d, otherwise start again with augmented dictionary.

4. if ($d \equiv 1 \ mod \ 8$ and $3 \not| (d)$ then Continue, otherwise increment j and Go to 3.

5. Call Cornacchia$(2^n, d)$.

6. If Cornacchia() succeeds assign the solution to π otherwise increment j and Go to 3.

7. Check whether norm of $(1 - v_i\pi)$ is equal to $c * q$ or not for any of units v_i in $\mathcal{O}$, where $c \leq C$, q is prime such that $q - 1$ is B-nonsmooth and $(2^n)^{2\varphi(c)}) \not\equiv 1 \ mod(c * q)$. If all these conditions are satisfied then Continue otherwise Set increment j and Go to 3.

8. Compute all the primitive binary forms using pbqf_cln(d).

9. Compute the precision $Prec(d)$ as given in Section 4.4.

10. Choose Yui-Zaiger's class invariant u_3 and obtain new reduced precision $Prec'(d)$.

11. Set $f(x) = 1$;

12. for k=1 to h

    (a) Obtain N for computation of η_N corresponding to k^{th} binary quadratic form.

    (b) Compute $\eta(\tau_k)$ till first n terms with $Prec'(d)$ precision.

    (c) Compute the chosen class invariant $u_3(\tau_k)$.

    (d) Compute $f(x) = f(x) * (x - u_3(\tau_k))$ with above defined precision.

13. Round off the coefficients of f(x) to nearest integer.

14. Now, determine the j-invariant as 48th power of root of class equation.

15. Assign to a any element of trace 1 or 0 as $(c * q) \equiv 2 \ or \ 0 \ mod \ 4$ respectively and $b := j^{-1}$.

16. **End**

We will discuss the implementation results in Chapter 7. In the next section, we discuss construction of supersingular curves over $GF(2^n)$.

4.8 Construction of Supersingular Elliptic Curves over $GF(2^n)$

Here, we briefly discuss the procedure for construction of supersingular curves over $GF(2^n)$. For supersingular elliptic curves over $GF(2^n)$, the choice for selecting a curve is very limited as there are only 3 and 7 isomorphism classes for odd and even n respectively [Men93b]. The following table depicts all of these isomorphism classes with their order and degree of extension for MOV attack which is minimum integer such that $((2^n)^k - 1)$ is divisible by $\#\mathrm{E}(GF(2^n))$.

Curve E($GF(2^n)$)	n	$\#\mathrm{E}(GF(2^n))$	k
$y^2 + y = x^3$	$n \equiv 1 mod 2$	$2^n + 1$	2
$y^2 + y = x^3 + x$	$n \equiv 1, 7 mod 8$	$2^n + 1 + 2^{(n+1)/2}$	4
	$n \equiv 3, 5 mod 8$	$2^n + 1 - 2^{(n+1)/2}$	4
$y^2 + y = x^3 + x + 1$	$n \equiv 1, 7 mod 8$	$2^n + 1 - 2^{(n+1)/2}$	4
	$n \equiv 3, 5 mod 8$	$2^n + 1 + 2^{(n+1)/2}$	4
$y^2 + \gamma y = x^3$	$n \equiv 0 mod 4$	$2^n + 1 + 2^{n/2}$	3
	$n \equiv 2 mod 4$	$2^n + 1 - 2^{n/2}$	3
$y^2 + \gamma y = x^3 + \lambda$	$n \equiv 0 mod 4$	$2^n + 1 - 2^{n/2}$	3
	$n \equiv 2 mod 4$	$2^n + 1 + 2^{n/2}$	3
$y^2 + \gamma^2 y = x^3$	$n \equiv 0 mod 4$	$2^n + 1 + 2^{n/2}$	3
	$n \equiv 2 mod 4$	$2^n + 1 - 2^{n/2}$	3
$y^2 + \gamma^2 y = x^3 + \beta$	$n \equiv 0 mod 4$	$2^n + 1 - 2^{n/2}$	3
	$n \equiv 2 mod 4$	$2^n + 1 + 2^{n/2}$	3
$y^2 + y = x^3 + \delta x$	$n \equiv 0 mod 2$	$2^n + 1$	2
$y^2 + y = x^3$	$n \equiv 0 mod 4$	$2^n + 1 - 2.2^{n/2}$	1
	$n \equiv 2 mod 4$	$2^n + 1 + 2.2^{n/2}$	1
$y^2 + \gamma^2 y = x^3 + \omega$	$n \equiv 0 mod 4$	$2^n + 1 + 2.2^{n/2}$	1
	$n \equiv 2 mod 4$	$2^n + 1 - 2.2^{n/2}$	1

Table 4.1: Representatives for Isomorphism Classes of Supersingular Curves

Here γ is a non cube in $GF(2^n)$ and $\lambda, \beta, \delta, \omega \in GF(2^n)$ are such that $Tr(\gamma^{-2}\lambda) = 1, Tr(\gamma^{-4}\beta) = 1, Te(\delta) \neq 0$ and $Tr(\omega) = 1$, where

$$Tr : \theta \longmapsto \theta^{2^0} + \theta^{2^1} + \ldots + \theta^{2^{n-1}},$$
$$Te : \theta \longmapsto \theta^{4^0} + \theta^{4^1} + \ldots + \theta^{4^{(n-2)/2}}$$

Let us assume that α is generator of multiplicative group of $GF(2^n)$, then following algorithm gives γ in $GF(2^n)$.

Gamma(α, n)

1. **Begin**

2. for $i = 1$ to $i = 2^n - 1$
   if $(i \not\equiv 0 mod 3)$
   then if $(i + 2^n - 1 \not\equiv 0 mod 3)$
   then if $(i + 2^{n+1} - 2 \not\equiv 0 mod 3)$ break;

3. $\gamma = \alpha^i$.

4. **End**.

Rest of the elements can easily be computed. For example, ω is an element with trace 1, $\alpha = \omega\gamma^2, \beta = \omega\gamma^4$ and δ is an element such that $Tr(\delta) \neq 0$.

With this we conclude this chapter. We will discuss the implementation results in Chapter 7.

Chapter 5

Elliptic Curves for Number Theoretic Computations

Apart from their application in cryptography, elliptic curves are also useful from the point of various number theoretic computations. In the recent past, several efficient algorithms have been proposed for certain number theoretic problems, i.e.integer factorization and primality testing. In this chapter, we briefly survey these algorithms and discuss their complexity.

5.1 Integer Factorization

The problem of factorization of composite numbers have attracted the attention of great mathematicians throughout the ages. Eratosthenes (250 B.C.) gave a method for determining all primes below a given limit. In the early 13th century, Fibonacci pointed out that to determine the factors of a number n, it suffices to trial divide by the integers $\leq \sqrt{n}$. Gauss emphasized the importance of the problem of distinguishing prime numbers from composite numbers, and gave methods for reducing the steps in trial division. In 1640, Fermat also gave an algorithm which works as follows.
Let $n = (x+y)(x-y)$. Starting from $y = 0$ till $y = \lfloor\sqrt{n}\rfloor$, test whether $n+y^2$ is a complete square or not. Once proper x and y are obtained, the factors will be given as $(x+y)$ and $(x-y)$.

More recently, the enhanced motivation for the study of the problem is the apparent security of the RSA public key cryptosystem based on the difficulty to factorize a number which is the product of two large primes. The issues related to computational complexity of various integer factorization algorithms have become very important. The above mentioned methods have the exponential complexity[1] $O(exp(1/2 \log n))$ and hence are impractical for

[1] $O(exp(c \log n))$ in general

large composite numbers. Though there is no polynomial time algorithm known for factorization, but recent progresses in this area have reduced the coefficient from 1/2 to lower values. In the present time, the quadratic sieve and number sieve methods are considered to be best and have been used to factor 129 digit composite number and some special forms of 155 digit composite numbers [Sim91]. The complexity of these method has been estimated as $O(exp((c+O(1))(\ln n)^a(\ln\ln n)^{1-a}))$. By giving some heuristic arguments $c = 2.08$ and $a = 1/3$ have been attained [Sim91].

For some special form of integers, other algorithms also give noticeably good result. Among these Pollard's $p-1$ method [Gre93, Ste85, PS93] was considered to be best. We give a brief description of this algorithm. Let M be a B-smooth integer, i.e. every prime factor p_i of M will be less than or equal to B. Thus

$$M = \prod_{p_i \leq B} p_i^{e_i} \qquad with \;\; e_i \geq 0$$

The probability that a random number $M \leq x$ is B-smooth is approximately u^{-u} for $u = \log x / \log B$. Now, suppose that composite number n is such that it has a prime factor p for which $p-1$ is B-smooth. Define

$$M = \prod_{p_i \leq B} p_i^{\lfloor \log n / \log p_i \rfloor}$$

The exponents of p_i have been chosen so that we can guarantee that $p-1$ divides M. In particular, for any integer a with $\gcd(a,p) = 1$

$$a^{p-1} \equiv 1 \; mod \; p$$

and so

$$a^M \equiv 1 \; mod \; p$$

To find p then, the method first computes $d = a^M - 1 \; (mod \; n)$ for some random a (say 2); secondly it computes the highest common factor of d and n. This will normally be p unless there are other primes p_i dividing n for which the exponent of a mod p_i divide M.

The time for this algorithm is dominated by the time to compute a^m modulo n. This algorithm requires $O(B(\log n)^3 / \log B)$ modulo n multiplications. These modulo n multiplications can be performed efficiently using Montgomery scheme [Mon85] for modulo arithmetic. In view of the earlier remark about the rarity of the B-smooth numbers, this method is not worth implementing unless it is known that, for some small B, $p-1$ is B-smooth.

Lenstra's elliptic curve factorization method [Ste85] is based on the same idea but resolves this problem by using elliptic curve group over $\mathbb{Z}_n$. This method succeeds if the order of the curve is B-smooth. The advantage of this method is that there are large number of different curves E that can be tried, each with potentially different value of order of curve. Explicitly,

the method is as follows:

Procedure ECFACT(n)

1. Begin.
2. Choose a value for B and let $M = lcm(1, 2, \ldots, B)$.
3. Choose an elliptic curve E randomly with integer coefficients and a point $\mathcal{P}$ on the curve modulo n.
4. Compute $M\mathcal{P}$ modulo n using the formulae given in Chapter 3. If the slope m of tangent while adding two points is infinite, then prime factor p of n will be gcd of denominator and n. If slope computation does not fail, then Go to step 3 and choose a different curve E.
5. End.

The correctness of this algorithm can be verified from the fact that if point at infinity is attained while carrying out the computation of $M\mathcal{P}$ then the slope m for the line passing through the points being added at that time will have denominator which will not be relatively prime to n. In other words, denominator will have a factor of n as the factor. Numerous choices of elliptic curve order over $\mathbb{Z}_n$ make this method very attractive as compared to Pollard's method. The complexity of this algorithm is estimated as $O(exp(\sqrt{\log n} \log\log n)^{1+O(1)})$. In practice, however, this method also does not give as good results as it promises for very large composite numbers. The reason for this is the rarity of B-smooth numbers. A significant improvement *may* be achieved by constructing curves with B-smooth orders over $\mathbb{Z}_n$ using the method given in Chapter 4, though constructing curves with given n and order may require very long time. It should be noticed that all the arguments which were given for fields in Chapter 2, will also hold for ring $\mathbb{Z}_n$ is well. That is, if n splits in principal ideals in any quadratic imaginary field $\mathcal{K}(\sqrt{d})$ into principal ideals $\mathfrak{n}$ and $\mathfrak{n}'$ then

$$(n) = \mathfrak{n}\mathfrak{n}'.$$

Here $\mathfrak{n}$ and $\mathfrak{n}'$ will not be prime ideals. If $\mathfrak{n}$ splits in Hilbert (or ring) class field $\mathcal{H}$ into ideals $\mathfrak{N}_i$ then $\mathcal{O}_{\mathcal{H}}/\mathfrak{N}_i$ will be a ring as $\mathfrak{N}_i$ is not prime ideal. The number of elements in this ring will be norm of $\mathfrak{N}_i$ which will be equal to n. Hence reduction map will give an elliptic curve over $\mathbb{Z}_n$.

5.2 Primality Proving

Primality testing is the most flourishing field in computational number theory. Though the integer factorization is very diffcult but primality testing is relatively much simpler. For primality proving several probablisitc and deterministic algorithms are available till date. Probabilistic algorithms, like Rabin Miller Test [Gre93, BS96] etc, tell with certainty about the compositeness of any number but primality is not certain. These algorithms are much faster than deterministic algorithms.We first give the Lucas theorem [Gre93, BS96, AM93, PS93] which is core of many deterministic and probabilistic primality testing algorithms.

Theorem 5.1 *If there exists an a relatively prime to n such that $a^{n-1} \equiv 1 \bmod n$ but $a^{(n-1)/q} \not\equiv 1 \bmod n$ for every prime divisor q of $n-1$, then n is prime.*

For more optimal form of this theorem please refer to [Wun83]. Now, if n is to be tested for primality then we require prime factors of $(n-1)$ so that Lucas test can be performed. Now, we go by a downrun process [Wun83] in which we first find small factors of $(n-1)$ less than some limit (which is computationally easier) and then carry out the Lucas test for all the factors. If test fails then n will be composite, otherwise we proceed with primality proving of indecomposed factor of $(n-1)$. Hence, this will be recursive procedure in which the size of the integer to be tested for primality will reduce with each step. It should be noticed that success is not guaranteed in this process as at any stage it may happen that indecomposable factor is composite and contains two large prime as factors. Hence, in such a case, algorithm will fail because factoring such an integer will be very difficult. Hence, the success of this method is largely dependent on the fact that no $(n_i - 1)$ should come across in the downrun process which contains two large prime factors. Please refer to [Wun83] for more detail.

We will now see that an elliptic curve analog of this method overcomes this problem to a great extent. Let us first see GoldWasser-Kilian theorem [AM93].

Theorem 5.2 *Let n be an integer prime to 6, E be an elliptic curve over $\mathbb{Z}_n$, together with a point $\mathcal{P}$ on E and m and s two integers with $s|m$. For each prime divisor q of s, we put $(m/q)\mathcal{P} = (x_q, y_q, z_q)$. We assume that $m\mathcal{P} = \mathcal{O}$ and $(m/q)\mathcal{P} \neq \mathcal{O}$ for all q. Then, if p is a divisor of n, one has $\#E(GF(p)) \equiv 0 \bmod s$.*

A closer look reveals that this theorem is a result of discussion in previous section. We also have:

corollary 5.1 *With the same conditions, if $s > (\sqrt[4]{n}+1)^2$, then n is prime.*

Combining this theorem with Schoof's algorithm [Sch85] which computes the order $\#\mathrm{E}(GF(p))$ of the curve in $O((\log P)^{8+\epsilon})$ steps, we obtain the Goldwasser-Kilian algorithm.

Procedure GK(n)

1. Begin.
2. Choose an elliptic curve E over $\mathbb{Z}_n$, for which the numbers of points m (computed with Schoof's algorithm) satisfies $m = cq$, with c a small integer and q a probable prime.
3. If m satisfy the condition of the theorem and corollary with $s = m$, then n is prime, otherwise it is composite.
4. The primality of q is proved in the same way.
5. End.

We see that the problem of limited choices for factors in the downrun process in the method mentioned earlier is overcome, as there are plenty of numbers to try. It should be noted here that downrun process is primality proving of q, because for a given q this algorithm will prove the primality of n in second step. Succesive downrun will reduce the size of q and once this chain of primality proving terminates, the primality of all earlier q will be obvious. If it so turns out in any step that any q is not prime then in the previous step order m should be changed with some other probable prime q, which will continue the downrun process. An astute observation reveals that q in each step is taken to be prime because it, in a way, avoids the factorization. Had the factorization been a simple task, we would have gone with it for testing the condition of the above theorem and corollary.

The problem with GK is that Schoof's algorithm is quite expensive n time in actual implementations. If we, instead, use the curve construction algorithm with known order then this problem will also be rectfied. Hence, contrary to what we do in procedure GK(n), we begin with some particular order and then find the related Weierstrass equation. The modified algorithm is as follows.

Procedure ECPP(n)

1. Begin.
2. Set $i := 0, n_i := n$,FLAG=0;
3. While(FLAG=0)
   (a) Set $prime\_n = 0$ and Search a discriminant such that n_i is expressible as norm. Denote the solution as π.

(b) Find all the choices of the order for each unit in corresponding order[2]. If any order m_i is **not** factorable into $F_i n_{i+1}$ where F_i is completely known factor and n_{i+1} is probable prime, then Go to step 3.1 and try with some other discriminant; otherwise (i.e. proper factors are found) continue.

(c) Store i, n_i, F_i, n_{i+1}.

(d) if n_{i+1} is prime then Set FLAG=1.

(e) Compute the class equation and curve equation as dicussed in Chapter 4.

(f) Find a point $\mathcal{P}$ on the curve.

(g) Check the conditions of theorem and corrolary with $s = n_{i+1}$ and $m = m_i$. In other words, check $F_i\mathcal{P} \neq \mathcal{O}$, $m\mathcal{P} = \mathcal{O}$ and $s > (\sqrt[4]{n} + 1)^2$. If all these conditions hold then Set $i := i + 1, prime\_n = 1$ and Go to step 3; otherwise Go to one iteration back in this loop and try with some other m_{i-1} so as to have different n_i. If $i - 1$ is negative the Set $prime\_n := 0$ and Go to step 4.

4. If $prime\_n = 1$ then n is prime otherwise composite.

5. End.

The complexity of this algorithm has been shown to be $O((\log n)^{6+\epsilon})$.
In the next section, we discuss an algorithm for finding square root in $GF(p)$.

5.3 Square Root Modulo p

In [Sch85], Schoof discusses an algorithm which begins with finding Frobenius element for elliptic curve over some suitable extension of $GF(p)$ which has complex multiplication with an order of discriminant x where x is the integer for which square root needs to be found. In fact, there is no need to go by the Schoof's method which unnecessarily adds extra overhead. By the theory discussed in Chapter 2, we know that Schoof's interpretation were correct but not necessary. We give here an outline of the modified algorithm.

Let $t^2 \equiv x \ mod\ p$. Then, we try to look for a minimum integer k such that p^k splits into principal ideals in an order of discriminant x in the quadratic field $\mathcal{K}(\sqrt{square\ free\ part\ of\ x})$. In such a case, p^k can be written as

$$4p^k = a^2 - xb^2$$

Where $a, b \in \mathbb{Z}$. Hence, it is easy to see that $a^2 = xb^2 \ mod\ p$, which means that $t = a/b \ mod\ p$. The search for proper extension should be begun with $k = 1$. If p splits into principal prime

[2]ring in quadratic field

ideals in quadratic field $K(\sqrt{square\ free\ part\ of\ x})$, then a proper soultion will be found for $k = 1$. The Cornacchia's algorithm discussed in Chapter 4 can be used for finding the solution.

Chapter 6

On Efficient Implementation and Smart Card Design

So far we have discussed how a cryptosystem can be constructed using elliptic curves over $GF(2^n)$ and $GF(p)$, and how the curve should be selected to ensure the security under the known attacks. In this chapter, we concentrate over various issues related to software and hardware implementation. Here our major focus will be upon the efficient arithmetic in $GF(2^n)$ & $GF(p)$, and selection of field and elliptic curves suitable for smart cards. In the first section, we give an introductory overview of smart cards.

6.1 Introduction to Smart Cards

Over the last few years, the computers and internet have become essential part of daily life. This inevitable invasion has led to an increase in demand of secure information and financial transaction over an electronically connected network. Apart from the banking applications, pay channels, telephones etc also demand the security. All these has led to heightened demand of smart cards which are equipped with cryptographic algorithms. A smart card is a multipurpose, tamper resistant security device. It is very much similar to credit cards but also possess storage and processing capabilities and can perform various cryptographic operations. The smart cards offer an economic and convenient solution to the problems of user authentication and has got plethora of applications including banking transactions, areas of health, mobile telephones and pay channels.

Traditional financial cards are magnetic strips cards [Sim91, DVJ96] which does not have processing capabilities but can store small amount of data. The scope of such cards is limited to very few applications. The additional ability to compute and interact in a system gives smart cards access to powerful cryptographic algorithms and improves the flexibility, security,

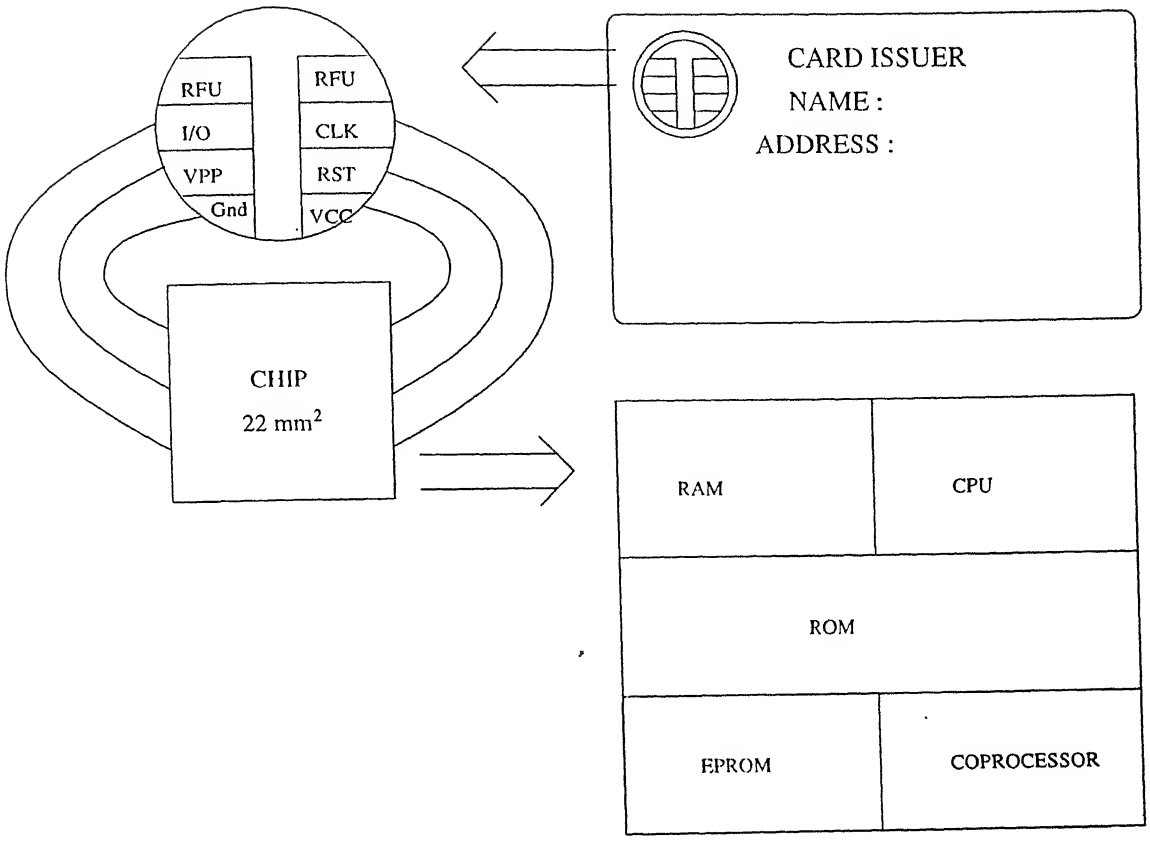


Figure 6.1: Component of Smart Card

and reliability of the system, or card. They make possible digital signatures, encryption, decryption and data access control. The advantage with the smart cards is that secret information never leaves the card in unencrypted form and moreover, each time encrypted data is different from what was used earlier. For more information on smart cards please see [Sim91, NM95, Kon91, dWQ99, FOM99, DVJ96, AMV93].

6.1.1 Components of Smart Card

The smart card looks like a credit card (see Figure 6.1). A rectangular plastic card supports all the electronic components. This plastic card also contains information concerning the application and issuer and also the information about the card holder. The following are the components of the smart card as specified by International Standards Organization (ISO 7816).

Contacts The smart cards contains eight contacts at the left corner as shown in Figure 6.1. The card communicates with external devices via a serial port (I/O in Figure 6.1). A common bit rate is 9600 bps but much faster rates (upto 115200 bps) are also used in full

accordance with standards. The contactless interfaces are also in practice which use RF signals to communicate. This feature increases the life of the smart cards. Other contacts are for clock signals, 5V power supply, reset and ground. The contact VPP is for programming the EPROM located on the card. The rest two contacts are not being used presently. Between these contacts, $22mm^2$ space has been provided for other electronic components.

Microcontroller The heart of the smart card is a microcontroller. The card's microcontroller executes the cryptographic application programs which are stored in ROM at the time of the manufacturing. It receives power and incoming data from external devices through contacts provided on the smart cards. Presently 8 bit microcontrollers are in use, the most common cores being Motorola's 68HC05 and Intel's 80C51. The development of 32 bit microcontroller is in progress for smart card applications [Cas94]. The 32 bit microcontroller will lead to significant improvement in the processing power of the smart card. Since the processing power of the card's processor is restricted due to several technological constraints, selection of a cryptographic algorithm is a very important issue. Presently RSA is in wide use for smart card applications, but we will see later in this chapter that elliptic curve public key algorithms are better choice for smart cards.

Coprocessor Since the public key algorithms are computation intensive, sometimes a coprocessor is also interfaced with the main processor to carry out the arithmetic operations involved in cryptographic protocol. The coprocessor is specifically designed to perform the required arithmetic efficiently. In RSA algorithm based smart cards, the coprocessor is designed to perform 512 bit integer modulo arithmetic. For cryptosystems based on DLP in $GF(2^n)$, the coprocessor is designed to perform arithmetic of 700 bit long elements. Since space available on the smart card is very limited ($22mm^2$), it is very difficult to design the coprocessor to perform arithmetic of such a large elements efficiently. Here, elliptic curves cryptosystem promise lot of scope as they require arithmetic of much smaller elements for the same level of security. Hence, the complexity of coprocessor is reduced. The size of the elements involved in arithmetic is so small (130 bits) that if the 32 bit microcontrollers are used then the main processor can be programmed to perform the required arithmetic, hence avoiding the need of coprocessor.

Memory The card contains RAM, ROM and nonvolatile memory (EPROM or EEPROM). The space requirement of memory is a critical issue in designing the smart cards as very little space ($22mm^2$) is available on the card. The typical values (in Bytes) for the size of these memories are 128, 3K and 4K respectively. The ROM contains smart card operating system written by mask during the chip manufacturing process. The RAM contains the intermediate results of the computations. The EEPROM contains the user-specific data individual to

each card. The EEPROM requires same power supply as that used for microcontroller (5V) and can be written or erased many times. The some portion of the EPROM is not accessible to external world which contains the secret information. Many precautions are taken while manufacturing the memory and other devices for smart card so as to avoid the possibility of the secret information getting leaked even if the cards gets into the hand illegitimate persons [Sim91].

Random Number Generator The random number generation is very important for any public key cryptosystem as we discussed in Chapter 3. This unit is not required if the random number can be generated by a software program. The security of a public key cryptosystem is largely dependent upon the randomness of the generated numbers. [Knu81] discusses various techniques for random number generation. Since in practice it is not possible to generate the perfect random numbers, there must be some mechanism incorporated to avoid the repetition of the sequence. One possible way is to update the seed for random number generation by a counter which changes the count with processing.

6.1.2 What Can A Smart Card Do?

The main purpose of the smart card is to authenticate the card holder to a system which is located at a remote place [Sch89, Miy92]. The smart cards are first initialized with proper keys so that a secure communication can be done using a cryptographic algorithm. The smart cards are inserted in a device called smart card reader (also electronic fund transfer machine in financial transaction). Once a smart card is inserted, the following authentications are done:

- User to Smart card: Every user is given a number called *personal identification number* (**PIN**). Using this PIN smart card identifies the user. The basic drawback with PIN is that it needs to be remembered and if it leaks out then the correct authentication will not be guaranteed. Instead of using PIN, biometric techniques, i.e. voice, finger print, are also used. But these techniques require more memory.

- Smart card to Remote System: Once the user verification is done, an authentication process is performed using cryptographic algorithms to prove the validity of card to remotely located system.

- Remote System to Smart Card: Similarly, a protocol is run to authenticate the system to card.

Once all of these processes are done successfully, the user is granted access to the system and further communication can take place. In the next section, we discuss a protocol for a

smart card in detail which uses an elliptic curve public key algorithm.

6.2 Elliptic Curve Based Smart Card

Here we discuss how an elliptic curve public key algorithm can be employed in a smart card to perform the tasks mentioned in the previous section. Various issues related to smart card design and efficient implementation will be discussed in the sequel.

If the data encryption is not required then Scnorr's scheme is a better choice to ElGamal's scheme as it requires less data to be transferred and the signature generation involves only one $k\mathcal{P}$ type of computation. The verification of signature generation requires two such computations but one of them requires less computations. Whenever the data is to be sent in encrypted form then ElGamal's algorithm can be used. Now we discuss the ElGamal's scheme for authentication only. Here no message will be involved and the process will be very much similar to Diffie and Hellman's key exchange protocol.

Let u & $\mathcal{U}$ $(= u\mathcal{P})$ and s & $\mathcal{S}$ $(= s\mathcal{P})$ be the private and public keys of the user and remote system respectively. $\mathcal{P}$ is the base point. The remote system maintains a data base for the public keys of the user. The following information is required to be stored in the card's memory.

- PIN number.
- Card's identity number.
- The field parameter. (p for $GF(p)$ and modulo polynomial for $GF(2^n)$).
- The coefficients of elliptic curve equation.
- The base point $\mathcal{P}$.
- The size (number of the bits) of the order of the elliptic curve.
- The secret key u.
- The public keys $\mathcal{U}$ and $\mathcal{S}$.

A typical example of financial transaction through verification by smart card goes as given below.

1. User to card authentication: User inserts the card in a smart card reader and enters his PIN. If the PIN matches with that stored in smart card memory, the process proceeds further otherwise terminates.

2. Card to system authentication: The card sends its identity number (not PIN) to system.

   (a) The system finds a random integer k, and computes the $k\mathcal{P}$. The $k\mathcal{P}$ is sent to the card.

   (b) The card computes $uk\mathcal{P}$ using the secret key and sends it back to system.

   (c) The system computes $k\mathcal{U}$ and compares with $uk\mathcal{P}$ received from the card. If the match is successful then the system can be sure of validity of the card.

3. System to Card authentication:

   (a) The card finds a random integer k, and computes the $k\mathcal{P}$. The $k\mathcal{P}$ is sent to the system.

   (b) The system computes $sk\mathcal{P}$ using the secret key and sends it back to card.

   (c) The card computes $k\mathcal{S}$ and compares with $sk\mathcal{P}$ received from the system. If the match is successful then the card can be sure of validity of the system.

Once the system and card authentication is done, the transaction can take place. The transaction data can be sent in encrypted (ElGamal's scheme) or plain text format (Schnorr's scheme) depending upon the requirement. The signature must always be put over each transaction data to ensure the authenticity throughout the process.

The microcontroller of the smart card is programmed to perform all the computations required in the above protocol. The application program is stored in either ROM or EEPROM. Apart from the security aspects, the encryption and decryption rate is also a major criteria for selecting any algorithm for cryptosystem. The throughput of any cryptosystem depends upon the complexity of the basic operation involved in encryption and decryption, which in case of RSA and elliptic curve cryptosystems are modular exponentiation and computation of multiple of a point ($k\mathcal{P}$) respectively. These operations are performed in terms of arithmetic of underlying field. The smart card often employs a coprocessor to carry out the computations involved in the public key algorithm. For RSA algorithms the coprocessor performs modulo exponentiation of at least 512 bit long integers. If an elliptic curve is selected over $GF(p)$ or $GF(2^n)$ such that the size of the field is of the order of 10^{40} (or greater) and all the conditions mentioned in Chapter 3 are satisfied then the corresponding cryptosystem will be secure. Since the elliptic curve public key algorithms require arithmetic of much smaller integers (or elements in $GF(2^n)$) for the same level of security, the complexity of the coprocessor reduces significantly.

The EEPROM contains all the information specific to user and algorithm, i.e. public key, private key, algorithm parameters. Sometimes it also contains the application programs.

Since the size of the available memory is limited, the storage requirements of any public key algorithm is also one of the major criteria for its selection for smart card.

Hence we see that in designing smart cards the storage requirements and the computational complexity of encryption and decryption are major issues. The computational aspects is handled by the coprocessor which is designed to perform the arithmetic in underlying field. Hence the efficiency of cryptosystem, in terms of implementations, is related to complexity and efficiency of the coprocessor. Now we focus our attention to various techniques for efficiently implementing elliptic curve arithmetic in hardware and as well as in software. Later on we will combine all the results to show the feasibility of employing elliptic curve public key algorithms in smart cards and compare them with other algorithms to show their superiority.

6.3 Efficient Computation of Multiple of a Point

As discussed in Chapter 3, computation of $k\mathcal{P}$ for a given integer k and a point $\mathcal{P}$ on elliptic curve is the basic operation in ElGamal's and Schnorr's algorithm. In this section, we discuss various techniques for efficient computation of $k\mathcal{P}$ in hardware and software implementation.

Since this computation is equivalent to exponentiation of integers, an analog of square and multiply [Knu81] for exponentiation, named accordingly as double and add, can be used. If

$$k = \sum_{i=0}^{t-1} k_i 2^i \qquad k_i \in \{0, 1\}$$

then

$$k\mathcal{P} = \sum_{i=0}^{t-1} k_i (2^i \mathcal{P})$$

Hence, we see that if k is a t bit integer then computation of $k\mathcal{P}$ requires $t-1$ doublings and at the most $t-1$ additions (not doubling). The number of additions is equal to number of ones in binary expansion of k. This algorithm is very useful for hardware implementation as it does not require any precomputations or extra storage. We will see in next section that doubling of a point in an elliptic curve group is much cheaper as compared to addition. Since in this algorithm, doubling is used to reduce the number of steps required in total computations, this algorithm becomes more attractive in view of above statement.

As we have seen in Chapter 3, the addition of a point is as expensive as subtraction because inverse of a point $\mathcal{P}$ ($\ominus\mathcal{P}$) can be computed at the cost of one addition in underlying field. Recognizing this very fact, we can improve the above algorithm by introducing a minor variation. This modified algorithm reduces the number of ones in the binary representation of k by using both subtraction and additions. In this algorithm the binary form k is rewritten as follows. Starting from the LSB side, bits are grouped in pair of two bits (as if k is written

with respect to base 4 with coefficients 0,1,2,3 in binary form). Now, whenever a pair of bits consists of two ones (coefficient 3 in base 4 representation), it is replaced by (0,-1) and 1 is added to next 2-tuple as carry. Whereas other 2-tuples, i.e. (0,0),(0,1),(1,0) are not changed. This process continues till MSB is reached. The example given below illustrates this clearly.

Example
Let $k = 98474747$. Then

$$k = \underline{01}\ \underline{01}\ \underline{11}\ \underline{01}\ \underline{11}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{01}\ \underline{11}\ \underline{11}\ \underline{10}\ \underline{11}$$

For this value of k double and add method will require 30 doublings and 21 additions. However, if we write k according to modified double and add method then we get

$$k = \underline{01}\ \underline{10} \overset{1}{\leftarrow} \underline{0\bar{1}}\ \underline{10} \overset{1}{\leftarrow} \underline{0\bar{1}}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{10} \overset{1}{\leftarrow} \underline{00} \overset{1}{\leftarrow} \underline{00} \overset{1}{\leftarrow} \underline{0\bar{1}} \overset{1}{\leftarrow} \underline{0\bar{1}}$$

Here, $\bar{1}$ represents -1 and $\overset{1}{\leftarrow}$ indicates flow of carry from right to left. Removing the arrows, we get

$$k = \underline{01}\ \underline{10}\ \underline{0\bar{1}}\ \underline{10}\ \underline{0\bar{1}}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{10}\ \underline{00}\ \underline{00}\ \underline{0\bar{1}}\ \underline{0\bar{1}}$$

Now, for the same value of k the $k\mathcal{P}$ can be computed with 30 doublings , 10 additions and 4 subtractions. Since, computationally subtraction and addition are same, it requires 7 less additions as compared to that required in double and add method.

If k is represented by a string of only ones, then this method shows great improvement because for $k = 2^t - 1$ simple double and add method will require $t - 1$ doublings and t additions, whereas modified version will require t doubling and 1 addition. In the worst case (k is represented by alternate 1 and 0's), the modified version will be same as simple binary method. Experimentally we have found that on average total number of additions required in computation of $k\mathcal{P}$ are one third of number of bits in binary representation of k. This method is suitable for hardware implementation as it does not require any precomputation results, except for modified form of k which is insignificant in comparison to total computations. In fact, the precomputation of the rearranged form of k can be avoided by using following approach.

Procedure Computekp(k, $\mathcal{P}$)

1. Begin.
2. $Sum = \mathcal{O}$.
3. while $(k > 0)$
   (a) $bit\_pair = k\ AND\ 3$.

(b) $k = k/4$.

(c) If $(bit\_pair == 3)$ then $Sum = Sum \oplus (\ominus \mathcal{P})$ and $k = k + 1$.

(d) If $(bit\_pair == 1)$ then $Sum = Sum \oplus \mathcal{P}$.

(e) $\mathcal{P} = \mathcal{P} \oplus \mathcal{P}$.

(f) If $(bit\_pair == 2)$ then $Sum = Sum \oplus \mathcal{P}$.

(g) $\mathcal{P} = \mathcal{P} \oplus \mathcal{P}$.

4. End.

Though, the above two method can be used for software implementation as well but the speed can be improved significantly by using other methods based on storage of precomputations. Since memory is not a problem in software implementation, the precomputation storage based methods are preferred for software implementations. Some of these algorithms are addition chain [Knu81], signed binary window method [KT92],vector addition chains [Roo95].

In the next section, we give the addition formulae in projective coordinates.

6.4 Addition Formulae in Projective Coordinates

From implementation point of view, it is desirable that for any cryptosystem the encryption and decryption must be simple and fast. We have already discussed the security aspect of elliptic curve based public key cryptosystems. In this section we explain alternative techniques for addition of points.

As we have seen in Chapter 3, the addition of two points requires computation of inverse of an element in underlying field. The computation of inverse is very expensive in terms of time. Here we give the addition formulae in projective coordinates [Men93b, AMV93, Miy92] to avoid the need of computation of inverse in every addition or doubling. An inverse computation will be required in computing the affine coordinates from projective coordinates and hence, only one inverse computation will be required in computation of $\mathcal{P}$. We adopt the following notations.
$\mathcal{P}, \mathcal{Q}, \mathcal{R} \in E$ and $\mathcal{P} = (X_1, Y_1, Z_1)$, $\mathcal{Q} = (X_2, Y_2, Z_2)$ and $\mathcal{R} = (X_3, Y_3, Z_3)$ with X_i, Y_i, Z_i are element of the field of definition.

For curves over $GF(p)$

Let the equation of the elliptic curve be given by

$$Y^2Z = X^3 + aXZ^2 + bZ^3 \quad where\ a, b \in GF(p)$$

The inverse of point $\mathcal{P} = \ominus\mathcal{P} = (X_1, -Y_1, Z_1)$.
If $\mathcal{P} \neq \ominus\mathcal{Q}$, $\mathcal{P} \neq \mathcal{Q}$ and $\mathcal{P} + \mathcal{Q} = \mathcal{R}$ then

$$\begin{aligned} X_3 &= vA, \\ Y_3 &= u(v^2 X_1 Z_2 - A) - v^3 Y_1 Z_2, \\ Z_3 &= v^3 Z_1 Z_2 \end{aligned}$$

where $u = Y_2 Z_1 - Y_1 Z_2$, $v = X_2 Z_1 - X_1 Z_2$, $t = X_2 Z_1 + X_1 Z_2$, $A = u^2 Z_1 Z_2 - v^2 t$.
If $\mathcal{P} = \mathcal{Q}$ then

$$\begin{aligned} X_3 &= 2uv, \\ Y_3 &= t(4A - v) - 8Y^2 v^2, \\ Z_3 &= 8v^3 \end{aligned}$$

where $v = Y_1 Z_1$, $u = t^2 - 8A$, $A = X_1 Y_1 v_1$, $t = aZ_1^2 + 3X_1^2$.

Here, we can see that while computing $k\mathcal{P}$ the inverse computation is avoided in each addition. The inverse operation is needed to compute the affine coordinates by substituting $x = X/Z$ and $y = Y/Z$. In the above expressions, number of multiplication can be reduced further if one of the Z-coordinate is taken to be one, i.e. affine coordinates. This can be done because in the computation of $k\mathcal{P}$, whenever 1 or $\bar{1}$ is come across either $\mathcal{P}$ is added or subtracted. Hence, if $\mathcal{P}$ is taken in affine coordinate then corresponding Z-coordinate can be taken 1.

For curves over $GF(2^n)$

Let the equation of the non-supersingular elliptic curve be given by

$$Y^2 Z + XYZ = X^3 + aX^2 Z + bZ^3 \quad \textit{where } a, b \in GF(2^n)$$

The inverse of point $\mathcal{P} = \ominus\mathcal{P} = (X_1, X_1 + Y_1, Z_1)$.
If $\mathcal{P} \neq \ominus\mathcal{Q}$, $\mathcal{P} \neq \mathcal{Q}$ and $\mathcal{P} + \mathcal{Q} = \mathcal{R}$ then

$$\begin{aligned} X_3 &= AD, \\ Y_3 &= CD + A^2(BX_1 + AY_1)Z_2, \\ Z_3 &= A^3 Z_1 Z_2 \end{aligned}$$

where $B = Y_2 Z_1 + Y_1 Z_2$, $A = X_2 Z_1 + X_1 Z_2$, $C = A + B$, $D = A^2(A + aZ_1 Z_2) + Z_1 Z_2 BC$.
If $\mathcal{P} = \mathcal{Q}$ then

$$\begin{aligned} X_3 &= AB, \\ Y_3 &= X_1^4 A + B(X_1^2 + Y_1 Z_1 + A), \\ Z_3 &= A^3 \end{aligned}$$

where $A = X_1 Z_1,\ B = bZ_1^4 + X_1^4$.
All the above mentioned argument hold here as well. If we compare this addition formulae with that for affine coordinates then we can see that in spite of few extra multiplication this is still cheaper.

For supersingular elliptic curve, the equation is given by

$$Y^2Z + aYZ^2 = X^3 + bXZ^2 + cZ^3 \quad \textit{where } a, b, c \in GF(2^n)$$

The inverse of point $\mathcal{P} = \ominus\mathcal{P} = (X_1, X_1 + Y_1, Z_1)$.
If $\mathcal{P} \neq \ominus\mathcal{Q},\ \mathcal{P} = \mathcal{Q}$ and $\mathcal{P} + \mathcal{Q} = \mathcal{R}$ then

$$\begin{aligned} X_3 &= AB^2Z_1Z_2 + A^4, \\ Y_3 &= B(A^2(Z_2 + A) + B^2Z_1Z_2) + A^2(Y_1Z_1 + aZ_1Z_2), \\ Z_3 &= A^3Z_1Z_2 \end{aligned}$$

where $B = Y_2Z_1 + Y_1Z_2,\ A = X_2Z_1 + X_1Z_2$.
If $\mathcal{P} = \mathcal{Q}$ then we need not go for projective coordinate as doubling operation requires inverse of coefficient a which can be precomputed. Hence, Z_1 and Z_2 can be taken to be 1. The X_3 and Y_3 are given by following expressions.

$$\begin{aligned} X_3 &= \frac{X_1^4 + b^2}{a^2}, \\ Y_3 &= \frac{X_1^2 + b}{a}(X_1 + X_3) + Y_1 + a. \end{aligned}$$

For supersingular curves, the doubling operation requires two multiplications and few squarings (which require just one clock cycle, see Section 6.6), and addition requires 10 multiplications. Hence the addition and doublings can be done more efficiently for supersingular curves over $GF(2^n)$ as compared to non-supersingular elliptic curves. But since the MOV reduction attack is a serious threat for supersingular curves, the curve and working field should be selected such that security of the cryptosystem remains unquestionable.

6.5 Efficient Arithmetic in $GF(p)$

In Sections 6.3 and 6.4, we discussed the efficient techniques for computation of $k\mathcal{P}$ and addition of points. Since additions requires arithmetic in underlying field, the efficiency of cryptosystem is largely dependent upon it. In this section, we discuss the implementation issues of basic arithmetic operations in $GF(p)$.

The basic arithmetic operations in $GF(p)$ is modular multiplication and addition. Other operations like square root mod p, inverse of an element etc can be computed in terms of

these. Hence, any improvement in these two basic operations will lead to enhancement in performance of overall system. In fact, the modular addition (or subtraction) is not much different then simple addition (or subtraction) except whenever the result exceeds the modulo integer p (or goes below p), p is subtracted (or added respectively) from it. Hence modular addition can be done at the cost of one comparison other than the simple addition.

But the multiplication modulo p is very expensive computationally because once the two integers have been multiplied, their modulo reduction can be done by trial divisions which are computation intensive operations. In 1985, Montgomery [Mon85] introduced a scheme in which the modulo multiplication can be done at the cost of one extra simple multiplication. We here discuss the Montgomery scheme for multiprecision arithmetic [DJ91] which is of our interest because the prime p will be of at least 130 bits.

This method requires, first of all, the problem variables to be transformed into a special p-residue form. To form the montgomery representation, we choose some R such that $R > p$ and relatively prime to p ($\gcd(R, p) = 1$). By choosing R to be some power of 2, division can be made inexpensive since division by any power of two means just right shifts. An integer x in Montgomery representation is given by

$$x_m = xR \ mod \ p$$

We translate normal integers to Montgomery representation, do our multiplications with this new representation, then translate back to the normal representation. Since R and p are relatively prime, these functions are one to one on $\{0,1,2,...p-1\}$. All these integers have a unique representation. To convert from Montgomery representation to normal representation, we find the inverse of $R \ mod \ p$ under multiplication mod p. Let it be R'. Then the inverse of x_m is

$$x = x_m R' \ mod \ p$$

In multiprecision case, the parameters p, R and the input X are multiple-precision integers and hence involve multiprecision arithmetic. That is, an integer x is represented as a sequence of digits $x_0, \ldots, x_{n-1}$ where

$$x = x_{n-1}b^{n-1} + x_{n-2}b^{n-2} +, \ldots, +x_1 b + x_0$$

and b is the base, typically a power of 2 (word size of the processor) and n is the number of digits. Also we choose $R = b^n$. The algorithm for modular multiplication is given as below. The result would be in Montgomery representation.

Procedure Montmult(A,B)
(** A and B are in Montgomery representation **)

1. Begin.

2. $p'_0 := -p_0^{-1} \ mod \ b$.

3. $T := 0$.

4. for $i := 0$ to $n - 1$ do

   (a) $T := T + A_i * B * b^i$.

   (b) $m_i := T_i * p'_0 \ mod \ b$.

   (c) $T := T + m_i * p * b^i$.

   (d) end.

5. $T := T/R$.

6. End.

Here, we can see that this method requires 2 multiplications to perform one modular multiplication, which is cheaper than trial and division based modulo reduction. The division by R in last step is simply right shift operation of bits, and hence can be done quite easily both in hardware and software implementation. [KABSK96] includes a comparison of few variants of Montgomery scheme for efficiency in time and space. Though the significant work is being done in developing hardware for Montgomery multiplication but it is better to write a module in assembly language over some DSP processor which can perform 32-bit multiplication in one clock cycle. If p can be represented by n blocks of 32 bits then this will require $(2n^2 + n)$ 32-bit multiplications. Hence the total number of clock cycles required will be of the order of $O(2n^2+n)$. In comparison to this, the number of clock cycles required to perform Montgomery multiplication in hardware equals the number of bits in binary representation of p. For example, if p is a 120 bit prime then hardware scheme will require 120 clock cycle whereas the DSP based implementation will require approximately 40 cycles.

Other than modulo multiplications, computation of square root modulo p and inverse of an element is also needed in cryptosystem implementation. We discussed one scheme for computing square root modulo p in Chapter 5. The inverse of any element can be found using extended Euclidean algorithm [Knu81].

All of above schemes involve simple multiprecision multiplications. Recognizing the fact that squaring is much simpler than multiplication, following observations leads to signific improvement. Let

$$m = \sum_{i=0}^{n-1} m_i b^i$$

Then

$$m^2 = \sum_{i=0}^{n-1} m_i^2 b^{2i} + 2 \sum_{i=1}^{n-2} m_i b^i \sum_{i=j+1}^{n-1} m_j b^j.$$

[Knu81] and [Zur94] contains several other algorithms for efficient multiplication of large integers.

6.6 Efficient Arithmetic in $GF(2^n)$

The finite field $GF(2^n)$ is very attractive for hardware implementation as its elements are represented by a string of n bits and concerned arithmetic operations are done using logical gates. The field $GF(2^n)$ is an nth degree extension of $GF(2)$ and is defined by a primitive polynomial of degree n over $GF(2)$ [Men93a, LN94, McE87]. Let $f(x)$ be such a polynomial and α be a root of this polynomial in $GF(2^n)$ then any $\beta \in GF(2^n)$ can be written as

$$\beta = \sum_{i=0}^{n-1} \beta_i \alpha^i \quad where \;\; \beta_i \in GF(2)$$

If α is known then β can be represented by the coordinates $\{\beta_i\}$ only. This representation is called standard basis representation. Obviously, the addition of two elements will give the third element whose coefficients will be bitwise modulo 2 sum of the coordinates of the two elements involved in summation. In other words, summation of two elements is equivalent to bitwise XORing of coordinates. The multiplication of two elements is equivalent to multiplication of two polynomials of degree $n-1$ modulo $f(\alpha)$.

Let $A = (a_0, \ldots, a_{n-1})$, $B = (b_0, \ldots, b_{n-1})$ and $C = (c_0, \ldots, c_{n-1})$ be their product. The following algorithm, which is suitable for both hardware and software implementations, can be used for computation of C. Here $f = (f_0, \ldots, f_n)$ denotes the coefficients of modulo polynomial.

Procedure GF2PROD(A,B)

1. Begin.
2. Set $C = (0, \ldots, 0)$.
3. for $i = 0$ to $n-1$
   (a) begin
   (b) if $(b_{n-1-i} = 1)$ then
   if $(c_n = 1)$ then $C = C \oplus f \oplus A$.
   else $C = C \oplus A$.
   else if $(c_n = 1)$ $C = C \oplus f$.

Then

$$m^2 = \sum_{i=0}^{n-1} m_i^2 b^{2i} + 2 \sum_{i=1}^{n-2} m_i b^i \sum_{i=j+1}^{n-1} m_j b^j.$$

[Knu81] and [Zur94] contains several other algorithms for efficient multiplication of large integers.

6.6 Efficient Arithmetic in $GF(2^n)$

The finite field $GF(2^n)$ is very attractive for hardware implementation as its elements are represented by a string of n bits and concerned arithmetic operations are done using logical gates. The field $GF(2^n)$ is an nth degree extension of $GF(2)$ and is defined by a primitive polynomial of degree n over $GF(2)$ [Men93a, LN94, McE87]. Let $f(x)$ be such a polynomial and α be a root of this polynomial in $GF(2^n)$ then any $\beta \in GF(2^n)$ can be written as

$$\beta = \sum_{i=0}^{n-1} \beta_i \alpha^i \quad where \;\; \beta_i \in GF(2)$$

If α is known then β can be represented by the coordinates $\{\beta_i\}$ only. This representation is called standard basis representation. Obviously, the addition of two elements will give the third element whose coefficients will be bitwise modulo 2 sum of the coordinates of the two elements involved in summation. In other words, summation of two elements is equivalent to bitwise XORing of coordinates. The multiplication of two elements is equivalent to multiplication of two polynomials of degree $n-1$ modulo $f(\alpha)$.

Let $A = (a_0, \ldots, a_{n-1})$, $B = (b_0, \ldots, b_{n-1})$ and $C = (c_0, \ldots, c_{n-1})$ be their product. The following algorithm, which is suitable for both hardware and software implementations, can be used for computation of C. Here $f = (f_0, \ldots, f_n)$ denotes the coefficients of modulo polynomial.

Procedure GF2PROD(A,B)

1. Begin.
2. Set $C = (0, \ldots, 0)$.
3. for $i = 0$ to $n-1$
   (a) begin
   (b) if $(b_{n-1-i} = 1)$ then
   if $(c_n = 1)$ then $C = C \oplus f \oplus A$.
   else $C = C \oplus A$.
   else if $(c_n = 1)$ $C = C \oplus f$.

(c) Give a right shift to bits of C.

(d) end.

4. End.

In software implementation, if elements of $GF(2^n)$ are represented by an array of binary word of x bits, then this algorithms computes product in $O(3n^2/4x)$ operations. The hardware circuit for above algorithm will require n clock cycles and the complexity of the circuit will depend upon number of non-zero coefficients in modulo polynomial $f(x)$. We will present the algorithm for the computation of inverse later. For computation of square roots in $GF(2^n)$ please see [McE87, Cha95].

Any element of $GF(2^n)$ can be considered as being n-tuples which constitutes an n-dimensional vector space over $GF(2)$. If $(\beta^{2^0}, \beta^{2^1}, \ldots, \beta^{2^{n-1}})$ is a basis for this space, then we call it a normal basis and β the normal basis generator.It can easily be shown that squaring of an element under normal basis representation is equivalent to one right cyclic shift of corresponding binary n-tuples. Hence if $A = (a_0, a_1, \ldots, a_{n-1})$ then $A^2 = (a_{n-1}, a_0, \ldots, a_{n-2})$. The addition is same as that for standard basis. The multiplication of two elements can be achieved by a logic function. Let $C = AB$ and logic function g gives the c_k from A and B. Then

$$c_k = g(a_0, a_1, \ldots, a_{n-1}; b_0, b_1, \ldots, b_{n-1})$$

Since

$$\begin{aligned} C^2 &= (c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \\ &= A^2 B^2 \\ &= (a_{n-1}, a_0, \ldots, a_{n-2})(b_{n-1}, b_0, \ldots, b_{n-2}) \end{aligned}$$

We get

$$c_{k-1} = g(a_{n-1}, a_0, \ldots, a_{n-2}; b_{n-1}, b_0, \ldots, b_{n-2})$$

Hence, we see that structure of the logic function required to compute any particular tuple of C is same. By giving cyclic shifts to vectors A and B, we can compute different bits of C.

It was proved in [MOVW89] that the logic functions g requires at least $2n-1$ logic gates. The normal basis which satisfies this bound is called *optimal normal basis*, ONB in short. Hence, with this representation, squaring can be done in one clock cycle and moreover, the complexity of the multiplier circuit will be minimum [MOVW89, GV95, ABV89, Men93a]. Obviously, the product will be obtained in n clock cycles. The ONB is very useful from the point of view of hardware implementation. See [Fen89, WTS+85, AMOV91] for various VLSI designs.

We will discuss the existence and construction of optimal normal basis in next section. Now, we give the method for the computation of inverse [Men93b]. Let $\gamma \in GF(2^n)$, then

$$\gamma^{-1} = \gamma^{2^n-2} = (\gamma^{2^{n-1}-1})^2$$

Now, if n is odd then

$$\gamma^{2^{n-1}-1} = (\gamma^{2^{(n-1)/2}-1})^{2^{(n-1)/2}+1}.$$

If n is even then

$$\gamma^{2^{n-1}-1} = (\gamma)^{2(2^{(n-2)/2}-1)(2^{(n-2)/2}+1)+1}.$$

In both the case, computations of the type γ^{2^r-1} is required. This term is computed by recursively applying this algorithm. [Fen89] discusses a VLSI design for inverse computation for small n. The complexity of the circuit for this algorithm is very high for large value of n. For large value of n, we give a variation of it which requires more number of multiplications but easy to implement in hardware. Let $n-1 = gh$, then

$$2^{n-1} - 1 = 2^{gh} - 1 = (\sum_{i=0}^{g-1} 2^i)(\sum_{j=0}^{h-1} 2^{jg})$$

Hence

$$\gamma^{-1} = \gamma^{2^n-2} = (\gamma^2)^{(\sum_{i=0}^{g-1} 2^i)(\sum_{j=0}^{h-1} 2^{jg})}$$

We can see that this algorithm can easily be implemented in hardware as it does not require recursive operation. It takes $g + h - 2$ multiplications. We ignore the squarings as they require just one clock cycle. This method will require minimum number of multiplications if the difference of g and h is as small as possible.

From the above discussion, it is obvious that optimal normal basis representation makes $GF(2^n)$ attractive for hardware implementation. In the next section, we explain the selection criteria for working field and curve for an efficient and secure cryptosystem.

6.7 Selection of Field and Curve for Smart Cards

So far we discussed the efficient techniques to carry out the computations in an elliptic curve public key algorithm for hardware and software implementation. In this section, we concentrate on selection of the field and curve for smart card application. Our major concern will be to minimize the storage for a smart card employing elliptic curve public key algorithm. We will also see if selection of field and curve can reduce the total computations involved.

We first concentrate on storage requirements [Miy92] which is very important in hardware implementation. Let the selected field be such that binary representation of its element require n bits. An elliptic curve based smart card requires following information to be stored in the memory.

- working field parameter (for $GF(p)$ only)[n bits].
- curve coefficient a, b (and c for supersingular curves over $GF(2^n)$)[$2n$ bits].
- secret key [n bits].
- base point [$2n$ bit].
- public key of self and system[$4n$ bits].
- size (number of the bits) in the order of the curve.
- PIN and card identification number.

Let us first concentrate on elliptic curve over $GF(2^n)$. As discussed in the previous section that optimal normal basis are very attractive for hardware implementation of $GF(2^n)$ arithmetic as the squaring and addition can be performed in clock cycle and multiplier has the minimal complexity. Unfortunately, the ONB does not exist in every extension of $GF(2)$. We will discuss its existence later in this section. But if the ONB exists in $GF(2^n)$ then it will be unique and corresponding multiplier will also be unique irrespective of the modulo polynomial used for defining the field $GF(2^n)$. Once we know the ONB representation of any element of $GF(2^n)$ then the modulo polynomial will not be required. Hence, there is no need to store the modulo polynomial in the memory of smart card.

If a non-supersingular elliptic curve is selected then addition formulae require only coefficient a. Hence only coefficient a needs to be stored. As we discussed in Chapter 4 the coefficient a is zero if the order of the curve is congruent to 0 modulo 4. Hence in such a case the storage can be reduced further. The addition of two points on supersingular curves requires a and b. The size of the secret key will approximately be same as that of field. The base point requires storage of two $GF(2^n)$ elements. The two public keys require storage of 4 elements of $GF(2^n)$. It may not always be necessary to store the public keys in the smart card if all the keys can be stored in a common data base which contains all the public keys with certification [Sta95]. The storage requirement for the size of the order is insignificant. Similarly PIN and card's identity number are always required. Hence we see that an elliptic curve based smart card requires $8n$ bit or $9n$ bit of storage as the the curve is non-supersingular or supersingular. If the non-supersingular curves are so selected that coefficient a is zero then $7n$ bits needs to be stored. Now we discuss the existence of ONB in any extension $GF(2^n)$ of $GF(2)$. The following two theorems tell us about the existence of ONB in $GF(2^n)$.

Theorem 6.1 *If $n+1$ is prime and 2 is primitive element of $GF(n+1)$, the $(n+1)$th roots of unity in $GF(2^n)$ form the optimal normal basis.*

Theorem 6.2 *If* $2n+1$ *is prime and either*

1. 2 *is primitive in* $GF(2n+1)$, *or*

2. $2n+1 \equiv 3 \bmod 4$ *and* 2 *generates the quadratic residues in* $GF(2n+1)$

then for $(2n+1)th$ *root of unity* β *in* $GF(2^{2n})$, $\gamma = \beta + \beta^{-1}$ *will be the optimal normal basis generator.*

Now, the selected field $GF(2^n)$ should be such that n satisfy any of the conditions given in these two theorems. If the condition of the Theorem 6.1 are met then $n+1$ will divide 2^n-1. If α is the primitive element of $GF(2^n)$ then $\alpha^{(2^n-1)/(n+1)}$ will be the optimal normal basis generator. If the second condition of Theorem 6.2 is satisfied then 2^n-1 will be divisible by $2n+1$ and $\alpha^{(2^n-1)/(2n+1)}$ will be the element β in the Theorem 6.2. The ONB generator γ can be compute using the relation given in Theorem 6.2. Now it remains to explain the first part of Theorem 6.2. In this case the construction of ONB generator is very difficult for known $GF(2^n)$ because this problem is equivalent to embedding of $GF(2^n)$ in $GF(2^{2n})$ which is very difficult for large field. Hence, this case will be impractical for the cases of curve construction algorithm in which modulo polynomial is obtained from the algorithm. For the cases given in Theorem 6.2, it is possible to obtain the minimal polynomial of the optimal normal basis generator [Men93a] using Fibonacci recursive sequence.
Let $f_0(x) = 1$ and $f_1(x) = x+1$. Then $f_n(x)$ generated by the following sequence will the minimal polynomial for ONB generator in $GF(2^n)$.

$$f_t(x) = x f_{t-1}(x) + f_{t-2}(x) \quad t \geq 2$$

Now, if we select n such that it satisfies first part (for second part also) of Theorem 6.2 and it is composite then, a curve can be constructed over a field which is subfield of $GF(2^n)$. Hence in such a case, polynomial of smaller degree will have to be factored in $GF(2^n)$, and $GF(2^n)$ is defined by minimal polynomial of ONB. For the architecture of the multiplier circuit, please see [GV95, AMOV91, Cha95].

The squaring and addition in $GF(2^n)$ are equivalent to one cyclic left shift and XORing respectively, and hence can be easily be implemented in hardware. The inverse of any element can be computed using repeated squaring and multiplication as discussed in the previous section. Hence the inverse computation circuit will require a multiplier and squarer (shifter). By proper interconnections between multiplier and squarer circuits and controlling the number of squarings and multiplication through software, inverse computation can be done efficiently.

Therefore, the coprocessor to be used in the smart card will consists of modules for multiplier, adder and shifter. Another important issue is data transfer overhead. Since the

present microcontrollers contains 8-bit long data bus, the fields elements can be transferred to coprocessor in 8-bit blocks only. If the working field is $GF(2^{128})$, then it will take at least $2 * 128/8 = 32$ clock cycles to transfer the two elements involved in multiplication and, moreover, 16 cycles will be required to get the multiplier output from the coprocessor. To avoid the data transfer overhead, a register bank can be built in the coprocessor which will contain the intermediate results. It would be desirable if coordinates of point $\mathcal{P}$ in computation of $k\mathcal{P}$ are transferred to coprocessor in the beginning, and rest of the operation are controlled by main processors through some control signals. While the register bank in coprocessor will store all the intermediate results. Since the size of the elements involved is quite small as compared to those involved in RSA coprocessors, the complexity of the coprocessor circuit will be significantly less. Hence, the making of register bank in the coprocessor should not be difficult.

If the working field is $GF(p)$ then the storage requirements can be reduced if x coordinate of the base point is zero. This will also reduce the computation requirements. If the x coordinate of the base point is not zero then both the coordinates need not be stored. If x coordinate along with least significant bit of y coordinate is stored then y coordinate can be computed from the elliptic curve equation. Same also holds for public key. Hence, under such conditions the storage requirements will be $7b + 2$. Since for a secure cryptosystem prime p will be of approximately 130 bits, the total storage required will be 782 bits. As we discussed earlier that modulo multiplication can be computed at the cost of two simple multiple precision multiplications using Montgomery's scheme. Hence if 32-bit microcontroller is used in smart cards then modulo multiplication in a prime field of 130 bit long order will require $2(130/32)^2 + (130/32) \approx 36$ 32-bit products. If the microcontroller can perform 32-bit multiplication in one clock cycle then one modulo multiplication can be performed quite efficiently. This avoids the need of the coprocessor.

In this section, we discussed issues related to smart card design. In the next section we compare the elliptic curve based smart cards with RSA based smart cards.

6.8 Comparison with RSA

As we discussed in the previous section, the elliptic curve based smart card needs $8n$ or $9n$ bit of storage if non-supersingular or non-supersingular elliptic curves over $GF(2^n)$ are used. If n is 130 then storage requirements is roughly 1K bits or 1.1K bits. Similarly, for curve over $GF(p)$ the storage requirement is approximately 912 bits if the field prime is 130 bit long. Now we make an estimate of the storage requirement of RSA based smart cards for the sake of comparison.

As we discussed in Chapter 1, RSA cryptosystem requires at least 512 bit modulus. In

case of RSA, only storage requirements are two modulo integer and public key pairs and one secret key. Since each of these is 512 bit long, the total storage requirement is approximately 2560 bits which is more than double of what is required for an elliptic curve based smart cards. Similarly, for the cryptosystem based on DLP in finite field, the n will be at least 700 and hence the total storage requirement 4200 bits. Hence we can see that elliptic curve cryptosystem require less memory in comparison to other public key cryptosystems.

The complexity of the coprocessor for RSA algorithm is far more than that required for an elliptic curve based algorithms. The security aspects were discussed in Chapter 3. Now, we make an estimate of the expected throughput. For information about the speed of RSA (and other) based cryptosystems, please see [DVJ96, DJ91, VVDJ99, OSA99, FOM99, Sim91, dWQ99].

The major computation in an elliptic curve public key algorithm is the computation of multiple of a point. In case of elliptic curves over $GF(p)$ addition and doubling of points require 15 and 13 modulo multiplications respectively. If k is x bit long and has Hamming weight y (number of 1's and -1's), then total number of multiplications required will be $15x + 13y$. Since for a cryptosystem to be secure, the order of the curve (and hence value of p as well) should be 130 bit long; the typical value of x and y will be 130 and 130/3. Hence total number of 130 bit modulo multiplications required will be approximately 2300. An equivalent cryptosystem with 512 bit long modulo integer requires 750 modulo multiplication of 512 bit long integers. If integers are represented in an array of 32-bit long words then each 512 bit long integer will requires 16 such words, whereas 130 bit long integer will require only 4 such words. Obviously, if coprocessor employs a 32-bit multiplier then computation of $k\mathcal{P}$ will require approximately $2300 * 16 = 36800$, 32-bit multiplications and RSA will require $750 * 16^2 = 192000$, 32-bit multiplications. These estimates clearly depict the superiority of elliptic curve cryptosystems.

Similarly, the computation of $k\mathcal{P}$ for non-supersingular elliptic curves over $GF(2^n)$ will require $7x + 13y$ multiplication because addition and doubling of points requires 13 and 7 multiplications respectively. Since each multiplication requires n bits, the expected number of clock cycles in computation of $k\mathcal{P}$ will be $13nx + 7ny$. If x and y are are 130 and 130/3 respectively then the number of clock cycles required in computation of $k\mathcal{P}$ will approximately be 202800, which ensures high throughput rate in comparison to existing RSA cryptosystems and other public key cryptosystems. The supersingular curves gives much higher throughput as number of multiplications in addition and doubling are 10 and 3 respectively.

Hence we see that elliptic curves are a suitable candidate for smart card applications. With the pace of the progress in computational technology, it is assumed that RSA with 512 modulo integer will face a serious security threats in near future. The RSA cryptosystems with larger modulo integers are also being realized. In such a scenario, the elliptic curve

cryptosystems will definitely be a better alternative for RSA cryptosystems.

Chapter 7

Implementation Results

Based on the theory discussed so far, we have developed a software package for elliptic curve public key cryptography which includes the programs for construction of elliptic curves and public key algorithms for cryptographic operations, i.e. encryption, decryption and signature generation. In this chapter we discuss various examples to cover all the cases mentioned in curve construction algorithms. In the first section, we give brief description of the implementations done.

7.1 Software Implementation

In this thesis our major concern is with the construction of elliptic curve over $GF(2^n)$ and $GF(p)$ which are suitable for cryptography. We also discussed various issues related to efficient implementation of cryptosystems. The thesis also includes software implementaion of elliptic curve public algorithm over $GF(2^n)$ and $GF(p)$. As the implementation of the elliptic curve public key cryptosystem require arithmetic in $GF(p)$ and $GF(2^n)$, various efficient routines have been developed to perform arithmetic of multiple precision integers and $GF(2^n)$. For the curve construction, we used a package called SIMATH which contains numerous routines to perform arithmetic in number fields and finite fields. Here we give a listing of the software implementation done. All the program have been developed in C-language and tested on **Pentium-100MHz** machine.

- Routines for multiprecision arithmetic: Several routines have been developed to carry out the modulo arithmetic for very large integers. The routine include integer multiplication, division, exponentiation, Montgomery modular multiplications and exponentiation, trial division modular arithmetic, gcd computation, inverse and modulo square root computation etc. Apart from these, routines for finding binary quadratic forms and class number for negative discriminants have also been developed.

- Routine for $GF(2^n)$ arithmetic: Various routines have been developed for $GF(2^n)$ arithmetic. Few of these are as follows. Product in standard basis representation, addition, inverse computation, square root computation, inversion of binary matrix for transforming the basis of representation.

- Curve construction programs: The algorithms discussed in Chapter 4 for construction of non-supersingular elliptic curves over $GF(p)$ and $GF(2^n)$ have been implemented on SIMATH, a package for algebraic computations. A program has also been developed for constructing supersingular curves over $GF(2^n)$. We will discuss the implementation results in this chapter later.

- Implementation of Cryptosystem: Various programs have been written to implement ElGamal's algorithm based cryptosystem over $GF(2^n)$ and $GF(p)$ using the constructed curves. The programs include the subroutines for finding random point on a curve, addition of two points, multiple of a point, generation of base point.

In the next section we discuss various examples of curve construction over $GF(p)$.

7.2 Construction of Non-Supersingular Elliptic Curves over $GF(p)$

In this section we give several examples to illustrate the performance of curve construction algorithm over $GF(p)$. As discussed in Chapter 4 we make use of a dictionary of negative discriminants and corresponding class numbers for search of discriminants. The dictionary that we used contained all the discriminants lying between -3 and -1000000. The size of the corresponding file is 11MB.

Example 7.1
Inputs

- Number of digits in prime factor of order $n = 75$,
- Lower bound on MOV extension attack $B = 30$.
- Upper bound on small factor $C = 100$.

Outputs

- The prime of $GF(p) =$
  $p = 14113600780497910708894654251027968028989202298238531961859572691369300585 73.$

- The coefficients of the constructed elliptic curve equation $y^2 = x^3 + ax + b$ are
  $a = 166814466418971645304389942292402701947982810259664927054121250247129053404,$
  $b = 656391562024722703850719637049242581469363392277328218922517731060326865 87.$

- The order of the curve $\#E(GF(p)) = c * q$
  $= 14113600780497910708894654251027968029525455802268563816845998638434246450 46$
  $= 2 * 705680039024895535444732712551398401476272790113428190842299931921712322523.$
  Here, the prime factor q is such that $(q-1)/2$ has no factor less than 30.

- j-invariant of the curve is
  $3253604797751353231094116934452593442595048687619085321253225834046549347 37.$

Now we give the intermediate results. As discussed in Chapter 4, the construction algorithm begins with determination of a prime factor q of the order of the curve such that $(q-1)/2$ has no factor less than 30. The determination of prime factor took 66 seconds. After this a negative discriminant is searched such that second norm equation has a solution and first norm equation gives the field prime p. In this example, the discriminant -43828 gives the proper solutions. The search of the discriminant took 607 seconds. The class number of this discriminant is 30, hence the degree of the class equation will be 30. For this discriminant all the primitive binary quadratic forms are computed using algorithm pbqf_cln(d). This discriminant satisfies the conditions for u_2 class invariant. The precision required ($Prec'(d)$) for the computation of class equation is 225 decimal digits. With this precision, the following class equation is obtained.

$f(x) = x^{30} + 509756793991686742315462149054112472459529433819596406039860251390317777285x^{29} + 74884827051070558223955300313255900945974246372513049576344466094795264029 6x^{28} + 1066204495441967637652121654372124919535102103081959616768345969666506796378x^{27} + 201279415341608830446423619837378880027707133859247852645351594234551916007x^{26} + 1056985106126799704462797522323302612631138548660689832926754343129020593533x^{25} + 1204559426566175057240533078281266058656584718472830900013496869190600655717024x^{24} + 4226923706457558583583083210256520889420571382171786451897350434172905048 87x^{23} + 49004580037108108192152744498524422915794976673716293841351915358526036816 3$

x^{22} + 59311154611671040571252035621274588654552957685002983145955689469869874195915955760094452345821593706795923867374436370538883929105869401 49x^{20} + 10641585426851223756196051188342881096834653893655003882891017007706633371 4x^{19} + 1127248956077789187469477687692276527511859593175827462524292762260020750471x^{18} + 95627420790848459323015692206259010003407048154871430550537583444635796022 0x^{17} + 64729983630988476525730462931276399908887192950514368619090710853087953575 7x^{16} + 41717364282309760898077955851113037010087133394557010729681494630786428230 4x^{15} + 5586956303990728803667142123536189581767131585186004711754935935439281212 40x^{14} + 1232296544039976281709858311666222188734606945607472321765384005276687401146x^{13} + 74487205054050777868803218695310753942692247993821131684572081050986172020 2x^{12} + 11896307374476369865786458210030991653187681763244644549860422431946053551 63x^{11} + 389768674469651138907543402192150708261855912124755997265175120400792805397x^{10} + 31572408677611168780607420252883597693887993569264678166779515244968026858x^{9} + 7350750694415685012297761691260301126186108785537005312326073927 64781982464x^{8} + 7493475302331353716995107227493056934293318868349358869533855744239004680 13x^{7} + 6877913525490771297383557748389012097733400610079768211603034915695452603 4x^{6} + 5523456496215614619420668163263031752427048583605937805476302903379707929 27x^{5} + 3272829434827797707558864367937558873795917385242259044330746494738981316891x^{4} + 2885723321529294984915610167236955376786983360304935613667655348228130802 23x^{3} + 1380537847202007432568534583642287866450926267526240227289640403184044246 643x^{2} + 1044422908857553733863812262696758819105343506247321326076390688443215023 686x + 997523233029604727386407662334625971534790413224756006595802385313234309779.

This polynomial gets factored completely over $GF(p)$. Its factorization was completed in 411 seconds. Its roots are as given below.

1001311506487213497138135897874527865659108669688921781723329720726996964646,
1290822594519257953264628714810706396558549940673684379299443418731586939883,
3955604740085836335814964408502999468600332635100163526558278234931269269055,
1301078127664482989500092461779593223043932073597772003834631660975630256378,
1223798172151249602364493979381495943106050305499489193781527508298422199531,
1371866729667131487768655857061433543350864283473439064374062481015270571 32,
9392196933182838492321058624071217749131885110401239397159546980457368350 1,

864361627070546886432544056064431147027687722088729397933298019985842892867,
132558004645959477999999282802165107207348098761575280120661168232472168874 5,
627485759157521131181180268612292451415039950199039876717991871655369234595,
140952396330130480917925003498669014369966320545269211245005022989203976349 7,
234112463379586053895895242166562764980776850362213977390185851329796268926,
804713011732474168926087448644893581301654914742726171532057916288748843660,
834590790615413791160461537753220638142740674983333911970263619394452748993,
106790564641028641247621627095246001324240932347627433556619003815874264388 7,
120711225189863816938075743571018621469077207857341247741042312151388218257 3,
583295232224504923029760401895110829515584668897331959072576871530931238 14,
105244041982582912112909204071409060743813339396395650720161045493723303720 7,
636447957259301909102122964884878639268563052607546523530669492899486849993,
266009492722703278074592476208586268103800283120985622758863859823463200755,
748431213156394204472205952458486738561273075611632970554609036175771014956,
830516616314131149485501323599379460960138215031179462072342523726762799907,
117120727427632848584422058070930727044078739272666498854882228079703927717 0,
795390803084964588190070257793077686488651246170185170727014390844626569077,
651961508554097131078641704531846950505328455150483449621109533875298422092,
478724668791374439539328296722675661889000027437586935082165233162737194021,
922411493727234339119707490899853495089014267653322088100000002652890839227,
814518762954712181668036662251071204838699412534425556259679119473616503789,
449121087506497680918183294852895220577458320383528977550070064352094537635,
788788634304047049699462202321905956081403623989261407326365442949777210944.

The above given j-invariant is 3rd power of first root of this polynomial in $GF(p)$. The other j-invariants can be computed in a similar way. The coefficients of the elliptic curve equation are obtained from j-invariant. For a given j-invariant over $GF(p)$ there are more than one isomorphism classes (2 for discriminants less than -4). To find the correct coefficients, a point $\mathcal{P}$ of high order is determined (randomly). If $\#E(GF(p))\mathcal{P}$ is point at infinity then the coefficients will be correct, otherwise correct coefficient are obtained as given in Chapter 4.

Example 7.2
Inputs

- Number of digits in prime factor of order $n = 40$,
- Lower bound on MOV extension attack $B = 1000$.
- Upper bound on small factor $C = 100$.

Outputs

- The prime p for $GF(p) = 477482957906613207189578608005296139839 3$
- The coefficients of the constructed elliptic curve equation $y^2 = x^3 + ax + b$ are

$$\begin{aligned} a &= 137252637900928823673155259672674430691 4, \\ b &= 431100684339971928535670040204144324405 2. \end{aligned}$$

- The order of the curve $\#E(GF(p))$ $(= c * q)$

$$\begin{aligned} \#E(GF(p)) &= 477482957906613207177098377587310630888 6 \\ &= 2 * 238741478953306603588549188793655315444 3. \end{aligned}$$

  Here, the prime factor q is such that $(q-1)/2$ has no factor less than 1000.

- j-invariant of the curve is 2943414941208503730054448587155247829400.

In this case, the prime factor is computed in 6 seconds. The discriminant is -12148. The class number of this discriminant is 18 and it was searched in 65 seconds. The precision $Prec'(d)$ is 126 digits. The class equation is as given below.

$f(x) = x^{18} + 13028802039808239212677724624315607345 4x^{17}$
$+ 18927173663845650755533676038741356832 7x^{16}$
$+ 45983359930222360644435216350499585708 67x^{15}$
$+ 80418118704352494242669138767774702123 6x^{14}$
$+ 22863399197597324346122382702425713489 94x^{13}$
$+ 24729719984697717339126624504765201217 84x^{12}$
$+ 46602084995445606047113811080918434657 61x^{11}$
$+ 20728367780060280168454848386266532345 9x^{10}$
$+ 21981667646975862110927111623583310023 20x^9$
$+ 19352937662090834480111316158993792022 74x^8$
$+ 24855149467637140301414776625990077238 8x^7$
$+ 99790397860594076054673634521999976807 7x^6$
$+ 22270294260387378347063920626353714207 06x^5$
$+ 16806621994332358969097388853633049735 62x^4$
$+ 16912663512140728057273661590442182421 43x^3$
$+ 98973912319570407003389554237465396049 7x^2$
$+ 30761423592263272265707679356514423193 13x$
$+ 35652630339520843603326684261962986293 74.$

This polynomial was factored in $GF(p)$ in 45 seconds. Its roots are as given below.

41642819656137875825457435711024809680 97,
81754370525288653578535213632253235374 2,
58413383523088732282101862236456223016 8,
19093001072879357341139881776633284007 99,
37581469527268809900649874479844831696 04,
10810414330231356773908389790169013235 80,
13897066725344940250852863518887356084 9,
88274934644522656819786709287367311347 1,
36704032679724132626480604325514390557 0,
37148356801887422222605861575042921358 77,
32002167293717602578724844951811821148 86,
50837485777940938236418993067966569004,
13760394236000484712854107768003257523 76,
28093766193832185453568153691934497823 15,
41917008760759335115145124861655625117 77,
25248719128090541468988199143904044291 66,
16575689290130735620810687854724783642 11,
64863037213639315944477630579933029805.

The j-invariant, given above, is obtained from the first root. The curve equation is computed in the similar way as discuss earlier. In this case, the degree of class equation is 18. In fact, the degree is not related to the size of the order as we illustrate in the next example that for the same size of the order, degree of class equation can be small. The degree of class equation (or equivalently class number of discriminant) depends upon particular prime factor but not on the size of the prime factor. It may happen that a large prime splits in an order of a quadratic imaginary field with small class number.

Example 7.3
Inputs

- Number of digits in prime factor of order $n = 40$,
- Lower bound on MOV extension attack $B = 100$.
- Upper bound on small factor $C = 100$.

Outputs

- The prime p for $GF(p) = 80158374625134291425290369651576836439 13$.

- The coefficients of the constructed elliptic curve equation $y^2 = x^3 + ax + b$ are

$$\begin{aligned} a &= 104716797799849689128648308454421348485 6 \\ b &= 337005780617014097503400104474870353787 5. \end{aligned}$$

- The order of the curve $\#E(GF(p))$ $(= c * q)$

$$\begin{aligned} \#E(GF(p)) &= 80158374625134291424170250440177319208 38 \\ &= 2 * 40079187312567145712085125220088659604 19 \end{aligned}$$

  Here, the prime factor q is such that $(q-1)/2$ has no factor less than 1000.

- j-invariant of the curve is 1630781539639094736437466886175741433795.

In this case, prime factor is generated in 30 seconds. The corresponding discriminants and class number are -436 and 6 respectively. The discriminant was found in 10 seconds only. Since the class number and discriminant both are small, the precision is 36 digits which is very less as compared to that in Example 7.2. In this case as well, u_2 class invariant is used for constructing class equation. The class equation is as follows.

$f(x) = x^6 + 80158374625134291425290369651545479732 45x^5$
$+ 80158374625134291425290367891287838709 53x^4$
$+ 80158374625134291425290125338574184776 41x^3$
$+ 80158374625134291425276671728929448642 65x^2$
$+ 80158374625134291425141627889371065533 53x$
$+ 80158374625134291423800127105904202439 77.$

Since the degree of this polynomial is very small, it was factored in 3 seconds. Its roots are as follows.

72868149619727881865342800469983307989 22,
78901288481751433222971954858608108785 13,
11951393919071593253916651258527328959 66,
65590909704541687576665618774252089207 78,
18859140900253972438011605304249273957 72,
72462615875190597344252847940718593563 69.

For each of these roots j-invariants can be computed. Here we have constructed curve for first root only. In the next example, the small factor of the order is taken different from 2.

Example 7.4
Inputs

- Number of digits in prime factor of order $n = 60$,
- Lower bound on MOV extension attack $B = 100$.
- Upper bound on small factor $C = 100$.

Outputs

- The prime of $GF(p) =$
  $p = 421481791575892841771242443084376956511882787740125618477727 9$.
- The coefficients of the constructed elliptic curve equation $y^2 = x^3 + ax + b$ are

$$a = 19508600258790851304953568087360867798010040810108849833290 23,$$
$$b = 15453455806520874485401566841008909477331249446784669395618 51.$$

- The order of the curve $\#E(GF(p)) = c * q = 421481791575892841771242$
  $443083973840105571538548541978094419 5 = 5 * 842963583151785683542484 88$
  $616794768021114307709708395618883 9$. Here, the prime factor q is such that $(q-1)/2$ has no factor less than 100.
- j-invariant of the curve is
  $604356003674505679413297494661885231336644307914555911861276$.

In this case, the prime factor is generated in 27 seconds. The discriminant is -5531 and it was found in 223 seconds. The class number is 23. Since discriminant is coprime to 6, except Yui-Zagier's class invariant all other class invariants can be used. If u_0 class invariant is used then required precision $Prec(d)$ is approximately 330. The precision $Prec'(d)$ for u_2 will be one third (=110) of it. Whereas for u_1 it is approximately 220 ($\approx (Prec(d) + 23\log|d|)/2 = (330 + 85)/2 = 208$). Hence, obviously the best choice is u_2. The class equation for all class invariants will be different but its degree and corresponding j-invariants in $GF(p)$ will be same.

$f(x) = x^{23} + 665587339001785099585356630743792 0x^{22}$
$+ 191812335548782935857932101529932124971296460 8x^{21}$
$+ 235491464538008756454558371268833649920033789276874047488x^{20}$
$+ 337242808071894370298247858753387882775422672942833637346853 0x^{19}$
$+ 386302754854146474403639531081291059820416608851278607381648 7x^{18}$
$+ 901634324943120961314912364267420716399259139195940767687947x^{17}$
$+ 363469865047349920698608488766772587326926566929924754667896 8x^{16}$
$+ 221919329743805410672469684422630801470571848889532313946255x^{15}$

$+ 1953912705370306460455937964184085131021470165317446279284658x^{14}$
$+ 4512184742443015563364748415130602302018520177870035253069 12x^{13}$
$+ 337846867926696280234029274200828059656376753841407276460049 4x^{12}$
$+ 253384599549731960921339607358744626341545443851481681383549 9x^{11}$
$+ 192998172256375730479161738738234434866189576209283895273267x^{10}$
$+ 100607860351660677953228555501232568951312439188784343654630 2x^{9}$
$+ 871513198266226312247965012038681228129254879352860006251146x^{8}$
$+ 279486391292459207580378605641670200313175695169726018154961 9x^{7}$
$+ 178427626103247349548955488768497057988211328446032953979063 2x^{6}$
$+ 262995097041449288945966294910592327849960054476394508745726 1x^{5}$
$+ 253735723004245289319426852819732788468207624050320743440321 0x^{4}$
$+ 383041648470920495127920224402023407393828921250152263392942 1x^{3}$
$+ 359377018677361979440819453607434159431336773859247992033529 9x^{2}$
$+ 674740671193324779896959455989408328313521893430774152954296x$
$+ 233920238041173326101873098156922092079825101389046278592844 8.$

This polynomial gets factored in 176 seconds. Following are the roots of this polynomial over $GF(p)$.

60603341529312731835369485683740726978737138269359570644692 2,
65884917194852219186111992070333458771233623545821694241534 8,
18596845590297812371780543474052693927468014970111075856235 22,
31566352983037641545422063323329726034795642156447911785129 89,
31957588934997743066787741265768313013790846611947453047622 90,
17542255161610220418768994224460527268700690296519991153476 76,
19903607995597348192706496474376935031523823608576221064244 77,
66517235734140390751603088330077385465924539456051976523195 3,
16402331482877111338713450739584640876160318651428177543387 80,
19695152229655886816530449582311297507693305707520364183410 25,
30789575542741182610229656999305035017641765679672111555455 46,
16150341128902870530719866863084798632806966293191383419503 55,
35561989141192462302680165372485193141278758632852875749395 37,
41826655731206554471078032285236436593255379989697786417694 88,
22331692752868335371611923793273364742084908512464283165167 87,
22402222493795206998395302695307146270239677404510730006206 24,
19740225805705903238960666249717283878308403796195015525184 92,
14649860467037588042050361569269160757321663868995119184500 3,
11960149571055860127406639829835329916857664599664128401108 76,

The roots of this polynomial are obtained in 10 seconds.
60518638794326884192484201009 77, 65799857188411333793327756555 65,
28869224931201174542213498041 48, 17594877791878836374372725413 28,
21390584128333530258339879270 04, 30606042543445262575056308842 47,
30985313959739944696738622923 50, 65474686666453794104713459733 72,
35275257834075814014218943544 65, 33363543166018735856650734308 03,
16160528473835925763428888316 81, 26233437988106037358310350537 46.

Let us summarize the above results.

Parameters	Example 7.1	Example 7.2	Example 7.3	Example 7.4	Example 7.5
d	-43828	-12148	-436	-5531	-3880
h	30	18	6	23	12
$\log q$	75	40	40	60	30
B	30	1000	100	100	120
c	2	2	2	5	20
$Prec'(d)$	225	126	36	110	81
t_p	$66s$	$6s$	$30s$	$27s$	$46s$
t_s	$607s$	$65s$	$10s$	$223s$	$33s$
t_r	$411s$	$45s$	$3s$	$176s$	$10s$

Table 7.1: Results of Examples

Here, t_p, t_s and t_r denotes the time taken in prime factor generation, search of discriminant and factorization of class equation. It is clear from this table that the discriminants with smaller class number are good from the point of view of computations. The maximum time taken in curve construction is 1084 seconds. But in this case, the order of the curve is also very large. Hence we see that non-supersingular elliptic curve can be constructed with predefined order quite efficiently.

Now, we discuss the particular case in which the order of the non-supersingular elliptic curve $E(GF(p))$ is p [Miy92]. As explained in Chapter 3, MOV attack will not be applicable over such curve. For this type of curves, the discriminant for which the two norm equations will have solution, will be square free part of $1-4p$. Now, instead of specifying p, we pick up a discriminant d such that it has small class number and check for a random integer x whether $(1-dx^2)/4$ is a prime or not. Here integer x is of the order of the square root of prime p (to be constructed). If $(1-dx^2)/4$ is a prime then assign this to prime p otherwise repeat the process for some other value of x. Obviously, success in this case will mean that two norm equations with $q = p$ and $\#E(GF(p)) = p$ will have a solution for discriminant d. Since

the discriminant is selected such that the corresponding class number is small, rest of the process will be simple. In our implementations we have found that finding such discriminant and prime is not difficult. In [Miy92], Miyaji discusses this idea for class number 1 only. He presented numerical results without much explanation. Here we will see that this concept fits well into the theoretical framework that we have developed. There is no need to restrict to the discriminant of class number 1, rather the argument holds for any class number. Of course, for small class numbers the computational complexity will be less as the degree of class equation will be small.

Example 7.6

Let $d = -11$ and field prime p be a 30 digit number. Hence we find a 15 digit number randomly and assign it to x. If $(1 - dx^2)/4$ is a prime then it is assigned to p otherwise process is repeated. An appropriate x was found in 0.43 seconds.

$$x = 274058689132959 \text{ and } p = 206547453995508613567745263123.$$

Since the class number of -11 is 1, class equation will be a polynomial of degree 1. $f(x) = x + 32$. Its root modulo p is 206547453995508613567745263091 and the j-invariant is 206547453995508613567745230355. The equation of the curve is
$y^2 = x^3 + 5452413136142809065204862402 5x + 47827947528849256669031045135.$
The whole process took just 2 seconds for constructing the curve.

Example 7.7

Similarly, for $d = -19$ and p a 60 digit prime, the prime p was obtained in 30 seconds. Here most of time was consumed in primality testing of p.
$x = 247138054970145802346623597745$
$p = 290116786518527339508643969064368625906962569595015953153869.$
The class equation is $f(x) = x + 96$. Its root and j-invariant are
290116786518527339508643969064368625906962569595015953153773 and
290116786518527339508643969064368625906962569595015952269133 respectively.
The equation of elliptic curve is
$y^2 = x^3 + 25109523043708798975017138842997986335807286725182667290 5100x$
$+ 16739682029139199316678092561998657557204857816788444860340 0.$
This curve was constructed in 61 seconds.

Example 7.8

If $d = -35$ and p is 60 digit number then prime p is obtained in 43.07 seconds.
$x = 102839685029476099538785 09$
$p = 134844010416263594946808966087824249091569213380428 3.$

In this case the class number is 2 and hence the degree of class equation is 2.

$f(x) = x^2 + 1348440104162635949468089660878242490915692133273067x$
$+ 1348440104162635949468089660878242490915691930845499.$
The roots of this equation are
1405292377267070831903624842502207758952585583305 75
1207910866435928866277727176628021715020433576004924.
The j-invariant corresponding to the first root is
65341295924166261152973576347174340942851373620 4282
and the equation of elliptic curve is
$y^2 = x^3 + 1281684927066873923923399792973113337256441537586373x$
$+ 9448820018149481782547813695957854453198583719715 22.$
The program took 7 seconds to construct the curve.

Example 7.9
If $d = -59$ then its class number is 3. If p is 40 digit prime then
$x = 9395424373451778963$
$p = 1302041487569463361021037060783787991693.$
The class equation is $f(x) = x^3 + 3136x^2 + 68608x + 720896$ and its roots are
1173008702346216304272655744874435180527,
6449328019023062524481317933101010940 04,
7861414708904041653212865833830397057 19.
The j-invariant is 19205397143593091166329421484551937755 1 and elliptic curve equation is given as
$y^2 = x^3 + 1267596398128927129259323675353120222 75x$
$+ 6690812618748593259258721832678154397 73.$
The precision used in the computation of class equation is 18 digits.

In this section, we discussed various examples for algorithms developed in Chapter 4. The algorithms are quite efficient and generate the curves in a very short time. We also discussed Miyaji's construction of curves with their order equal to field prime in the general perspective. In the next section, we consider various examples of non-supersingular elliptic curve construction over $GF(2^n)$ for different cases given in Chapter 4

7.3 Construction of Non-supersingular Elliptic Curves over $GF(2^n)$

In this section, we discuss the results of the algorithm developed for construction of non-supersingular elliptic curve over $GF(2^n)$. In [LZ94], the similar algorithm have been discussed but it was discussed for one particular case. Whereas we have generalized that algorithm to include several other cases which were (possibly) discarded in [LZ94] due to incorrect interpretation of an important result concerning the prime ideals in class group (see Chapter 4 and Appendix C). Here we will validate all the arguments given in Chapter 4 by illustrating examples for each case.

Since for cryptographic applications, the size of the field has to be at least 10^{30}, the extension degree of $GF(2^n)$ over $GF(2)$ must be greater than 120. As we discussed earlier the extension degree is related to class number (equal in some cases), hence the class number for construction will be very high. This results in very high precision for construction of class equation. Hence to reduce the precision required, we will consider only Yui-Zagier's class invariant u_3 as it reduces the precision by a factor of 48. Moreover, one of the two conditions[1] on discriminant to use u_3 is a necessary condition for 2 to split into prime ideals in quadratic imaginary fields. Now we consider each case separately using the same notations as given in Chapter 4.

n is a composite number with h as a factor

Here $n = ht$. In this case, discriminant is searched in class number h and $q = 2^n$ in the norm equations. Obviously, the class equation will a polynomial of degree h which *may* reduce in $GF(2)$ if h is not prime. Let us first consider the case in which the class equation is irreducible in $GF(2)$.

Example 7.10
Inputs

- $n = 144$.
- $h = 36$.
- $B = 10$.
- $C = 100$.
- The modulo polynomial is (144 1000000000000000000000000000000000

[1] $d \equiv 1 \ mod \ 8$

000000000

Inputs

- $n = 128$.
- $h = 32$.
- $B = 10$.
- $C = 100$.
- The modulo polynomial is (128 10000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000110011011
  01100100111011111111).

Outputs

- The elliptic curve equation is $y^2 + xy = x^3 + ax^2 + b$, where
  a = (127 11001011000011111111011010100001010010101101011000111011111011011
  10100011010100001100100000100101011000100011001000000000010001100),
  b = (126 10000100100000010010010011001011110000000000010100100100111101
  01011011011000000000000101100010100010101010000001001100010111000).
- The order of the curve

$$\begin{aligned} \#E(GF(2^{128}) &= 340282366920938463494384599195483217314, \\ &= 2 * 170141183460469231747192299597741608657. \end{aligned}$$

- j-invariant= b^{-1}.

In this case, the discriminant is -4495. Here, integer 2 splits into prime ideals $\mathfrak{p}, \mathfrak{p}'$ which have order $h/2$ ($= 32/2 = 16$). Hence $\mathfrak{p}^{16}$ will be principal ideal in class group of $\mathcal{O}(-4495)$. This means that the class equation will get reduced to two irreducible polynomials of degree 16 in $GF(2)$. The precision required is 27. The class equation is

(32 11111101001101110010010111000010 1).

The polynomial (16 11010011010011001) is one factor of the class equation. Now, for computation of j-invariant this polynomial is factored in $GF(2^{128})$. The factorization is done in 1 hour and 15 minutes which is quite less as compared to that required in Example 7.10. Once root is obtained the coefficient b can easily be computed. The coefficient a is an element with trace 1. Similarly, the anamolous elliptic curves [MS99] can be constructed if h is taken 1 and n as the desired extension. Now, we discuss the case when $n = h$.

$\underline{n = h}$

First we discuss the case when n is prime.

Example 7.12

Inputs

- $n = h = 193$.
- $B = 10$.
- $C = 100$.

Outputs

- The modulo polynomial is (193 1101111111010101010001001100000111011000010111111110001000010100001111011010101110101011110001010000011101010100101001111110111010010011111011010100001001010100110110001101111100000011000010011)
- The elliptic curve equation is $y^2 + xy = x^3 + ax^2 + b$, where
  $a = 0$,
  $b = (192\ 111110000110111110111110011000100101111000001111111011101101010001010010001001101011001100101001001111000011011111001010001001000010011101010011111111111011111111110011001110110111100001011100111)$.
- The order of the curve
  $\#E(GF(2^{193}) = 12554203470773361527671578846415371030581749923957395864724 = 12 * 1046183622564446793972631570534614252548479160329782988727340282366920938463494384599195483217314$.
- j-invariant$= b^{-1}$.

The discriminant is searched in class number 193. A proper solution is found for discriminant -42407. Since the class number is a prime, every ideal in class group $\mathcal{CL}(\mathcal{O})$ will have order 193 and hence the factors of 2 as well. The class equation will be an irreducible polynomial (over $GF(2)$) of degree 191 and can be used as modulo polynomial for defining the field. If the root is α then α^{-48} will be the coefficient b. The coefficient a is zero as $\#E(GF(2^{193}) \equiv 0\ mod\ 4$. The precision used in computation of class equation is 63. Now we consider the case when $n = h$ and n is composite. Now there will be cases, first in which the class equation is irreducible and second in which the class equation reduces over $GF(2)$. We consider the first case first.

Example 7.13
Inputs

- $n = h = 160$.
- $B = 10$.
- $C = 100$.

Outputs

- The modulo polynomial is (160 1101000100111000001111111111111100101010100101010100010011111000101101001011110010101100100001000101101010110001110111000001100101110100101100110001111100100100).
- The elliptic curve equation is $y^2 + xy = x^3 + ax^2 + b$, where
  $a = 0$,
  b = (159 1011010100010000111110001011000101010101110000100011101111101111000000001110100001101011001010100110101011110000111110001100010011000110100111100101111000010).
- The order of the curve

$$\begin{aligned} \#E(GF(2^{160}) &= 1461501637330902918203687131986134054232807615724, \\ &= 4 * 365375409332725729550921782996533513558201903931. \end{aligned}$$

- j-invariant$= b^{-1}$.

In this case the discriminant is -20495. Obviously the factors of 2 in $\mathcal{O}(-20495)$ will be of order 160 and the corresponding class equation will be irreducible. This is used for defining the field. The precision required is 54. The coefficient a will be zero and b can be obtained by raising the inverse of the root of this equation to 48th power. The next example discusses the case when the class equation gets factored for this very class number.

Example 7.14
Inputs

- $n = h = 160$.
- $B = 10$.
- $C = 100$.

- The modulo polynomial is (160 1000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000001001000101111111010010110000011).

Outputs

- The elliptic curve equation is $y^2 + xy = x^3 + ax^2 + b$, where
a = (159 110010001101101111111000011110111010101011010101111100011 1
001011111000010110101001001110100111111011010111101101101011100101 1
100010100010110110001010000000001).
b = (159 111000100010010011011010101000100000000101111011100011100110
111111001100011100101111101000111100010000001101100100010111111010000
001001110000110001101101001111 00).

- The order of the curve

$$\begin{aligned} \#E(GF(2^{160}) &= 1461501637330902918203685262673156328315700024354, \\ &= 2 * 730750818665451459101842631336578164157850012177. \end{aligned}$$

- j-invariant= b^{-1}.

In this case the discriminant is -55279. The required precision is 63. The class equation is a reducible polynomial of degree 160 and each of its factor is a polynomial of degree 20. The class equation is given by
(160 1101001110101100111000001110111000001100110010110011101011 0
101010011001100101000011001011000101001001110001011010011011111101 1
001110001100000100101111010110000101).
Its one factor is (20 1000000101000101010001). The coefficient a will be an element of trace 1. For computation of b, this polynomial is factored. Its factorization is done in 4 hours. Now we discuss the last case.

$\underline{h = nt}$

Example 7.15

Inputs

- $n = 135$.
- $h = 270$.
- $B = 10$.
- $C = 100$.

Outputs

- The modulo polynomial is (135 1000111011111111010101011000101100011010000111001010010000000000000101101110000110011001010110111110010101110000001001101010100010111101).
- The elliptic curve equation is $y^2 + xy = x^3 + ax^2 + b$, where
  $a = 0$,
  b = (133 1001000011101111111101100100100100010110111110111101000000101001110101110111101010011010001010001111010101001100001101101000111010110).
- The order of the curve

$$\begin{aligned} \#E(GF(2^{160}) &= 4355614296588012332341403665512292735534 8 \\ &= 12 * 362967858049001027695116972126024394627 9. \end{aligned}$$

- j-invariant= b^{-1}.

In this case, the discriminant is searched in class number 270. Obviously, a solution will be found iff the corresponding ideals will have order 135. The discriminant is -48599. The precision used for computation of the class equation is 90. The class equation have two factor polynomials of degree 135. The equation of the class equation is give below.
(270 1000010011110110001101101100011111111101010111100101011
000010011100100010010100011110111111011010010011110010110000101111100100101001111000000100111000111100111100010000111000100101000011110000111010101000100100001111100000101110101101100100110101001000001).
Its one factor is (135 1000111011111111010101011000101100110100001110010100100000000000000101101110000110011001010110111110100101110000000100110101010001011110 1).
This polynomial can be used as modulo polynomial for defining the field $GF(2^{135})$.

Hence we see that all of above cases efficiently generate the elliptic curve over $GF(2^n)$. If an option is so chosen that factorization of the class equation is avoided then this algorithm constructs the curve in few second for fields with extension degree n smaller than 150. For larger field also this method gives the curve in few minutes. Now in the next section, we give the results for supersingular elliptic curve construction over $GF(2^n)$.

7.4 Construction of Supersingular Elliptic Curves over $GF(2^n)$

As discussed in Chapter 4, there are only 3 and 7 isomorphism classes of supersingular elliptic curves over $GF(2^n)$ for odd and even n respectively. For each class, the order of the curve is known. The representative reduced form for each isomorphism class of supersingular curves over odd extension of $GF(2)$ are given in Chapter 4. For even extension of $GF(2)$, the representative form for each of seven isomorphism classes can obtained by finding elements in $GF(2^n)$ following certain properties discussed in Chapter 4. To construct these curves, we have developed routines for large integer and $GF(2^n)$ arithmetic.

Here, we give examples for each isomorphism class.

Example 7.16
Isomorphism classes over $GF(2^{155})$

1. If equation of the curve is $y^2 + y = x^3$ then the order of the curve is 45671926166590716193865151022383844364247891969. The degree of MOV reduction attack is 2.

2. If equation of the curve is $y^2 + y = x^3 + x$ then the order of the curve is 45671926166590716193865151022383844364247891969. The degree of MOV reduction attack is 4.

3. If equaton of the curve is $y^2 + y = x^3 + x + 1$ then the order of the curve is 45671926166590716193865453253838748021541568513. The degree of MOV reduction attack is 4.

Example 7.17
Isomorphism classes over $GF(2^{160})$
The modulo polynomial for $GF(2^{160})$ is (160 100000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000100
10001011111110100101100000011).

1. If equation of the curve is $y^2 + (1\ 10)y = x^3$ then the order of the curve is 1461501637330902918203686041642102634285107249153. The degree of MOV reduction attack is 3.

2. If the curve equation is $y^2 + (2\ 100)y = x^3$ then the order of the curve is 1461501637330902918203686041642102634285107249153. The degree of reduction attack is 3.

3. If equation of the curve is $y^2 + (1\ 10)y = x^3 + (135\ 10000000000$
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000) then the order of the curve is
1461501637330902918203683623790463405026757836801. The degree of MOV reduction attack is 3.

4. If equation of the curve is $y^2 + (2\ 100)y = x^3 + (137\ 1000000$
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000) then the order of the curve is
1461501637330902918203683623790463405026757836801. The degree of MOV reduction attack is 3.

5. If equation of the curve is $y^2 + y = x^3$ then the order of the curve is
1461501637330902918203682414864643790397583130625. The degree of MOV reduction attack is 1.

6. If equation of the curve is $y^2 + y = x^3 + (133\ 10000000000$
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000)
then the order of the curve is
1461501637330902918203687250567922248914281955329. The degree of MOV reduction attack is 1.

7. If equation of the curve is $y^2 + y = x^3 + (1\ 10)$ then the order of the curve is
1461501637330902918203684832716283019655932542977. The degree of MOV reduction attack is 2.

The only computation intensive operations involved in constructing the supersingular curves are search of elements of trace 1, which too is accomplished within no time.

7.5 Implementation of Cryptosystems

The elliptic curve cryptosystems based on ElGamal's algorithm have been implemented in software. Various routines have been developed to perform the arithmetic of modular integer and elements of $GF(2^n)$ to implement the cryptosystem over $GF(p)$ and $GF(2^n)$ respectively. For modular arithmetic in $GF(p)$ Montgomery's modulo multiplication algorithm have been used. The following table shows the encryption and decryption rate over different finite fields.

Finite field $GF(q)$	$\#GF(q)$ in decimal digits	Encryption rate bits/sec	Decryption rate bits/sec
$GF(p),\ q = p$	51	400	1240
$GF(p),\ q = p$	30	790	1620
$GF(2^{130})$	39	320	733
$GF(2^{155})$	47	256	500

Table 7.2: Throughputs of cryptosystem over $GF(q)$

It is obvious from the data given in table that the throughput decreases as the size of the working field increases. For the fields with almost same order, the elliptic curve cryptosystem over $GF(p)$ have higher throughput in comparison to $GF(2^n)$ as 32-bit multiplication is very efficient on Pentium processor. The decryption rate is almost twice of encryption rate because encryption involves 2, $k\mathcal{P}$ type of computations. For the computation of $k\mathcal{P}$ modified double and add method has been used. Higher throughput can be achieved if precomputation based algorithms are used and routines for arithmetic in underlying field are written in assembly language.

Chapter 8

Conclusions and Future Work

After going through various issues involved in design of efficient and secure elliptic curve cryptosystems, we conclude the thesis with review of main points and scope for future work.

8.1 Conclusions

The selection of a suitable elliptic curve and the finite field is an important aspect in designing an efficient cryptosystem. From the implementation point of view, elliptic curves over finite fields $GF(p)$ and $GF(2^n)$ are of particular interest. We discussed the algorithms for construction of non-supersingular elliptic curves over $GF(2^n)$ and $GF(p)$ which are secure against all of known attacks. The algorithms are based on Lay and Zimmer's scheme (which is modified version of Atkin and Morain's scheme [AM93]) and have been generalized to include several other cases.

The algorithm for construction of non-supersingular elliptic curves over $GF(p)$ gives a generalized framework for construction of curves. The algorithm have been implemented and tested to generate the elliptic curves over $GF(p)$ with predefined order. Miyaji's method [Miy92] for construction of elliptic curves over $GF(p)$ with order p turns out to be a particular case. The algorithm have been tested to generate the curves with order p in few seconds for discriminants with class number other than 1 also.

In case of construction of elliptic curves over $GF(2^n)$, the Lay and Zimmer's algorithm have been generalized to include several other cases in which class number is not equal to the degree of extension of $GF(2^n)$ over $GF(2)$. These cases were excluded by Lay due to incorrect interpretation of irreducibility of class equation (possibly, as appears from one line written in [LZ94]). Whereas we present those cases as favorable ones when it is desirable to specify the modulo polynomial for $GF(2^n)$. As discussed in Chapter 6, this is required when optimal normal basis representation is desired. We support our argument with theoretical justification as well as by illustrating examples for each case. The algorithms have been

tested thoroughly. The superiority of these algorithm over Schoof's algorithm [Sch85] is obvious as the time required in construction of curves over very large fields is very less (few minutes on Pentium-100MHz for curves over field as large as $GF(2^{300})$) as compared to the time required by Schoof's algorithm [Men93b]. Apart from non-supersingular elliptic curves, supersingular curves over $GF(2^n)$ have also been discussed and implemented.

Thesis also discussed various issues involved in implementation of elliptic curve cryptosystems. Various aspects related to selection of curve and working field for smart card implementation have also been discussed. It is obvious from the discussion in Chapter 6 that elliptic curve public key algorithms are a better candidate for smart card than RSA as they are less expensive both in terms of memory and computational requirement for same level of security.

8.2 Future Work

In this thesis, we discussed various aspects of designing an efficient and secure elliptic curve public key cryptosystem. The task of constructing the curve have been fully accomplished which takes care of security aspects. This work can be extended to take up other aspect of an elliptic curve cryptosystem design, that is, hardware and smart card implementation. This will be more concerned with VLSI design related issues. This is a very challenging task as limited space is available on smart card and moreover, the processing power is also restricted.

The elliptic curves have been studied in detail in the thesis. Apart from their application in cryptography, the elliptic curves have also been discussed for primality testing and integer factorization which have always been a hot topic of research in number theory. As discussed in Chapter 5, the curve construction algorithms can be used to improve the efficiency of existing algorithms for integer factorization and primality proving.

Appendix A

Modular Forms

Here we give a brief overview of theory of modular forms. For details please refer [Sil94, Apo76, Seg80]. In the discussion k denotes an integer, $\mathbf{H}$ denotes the upper half-plane, $\mathbf{H}= \{\tau : \Im(\tau) > 0\}$.

Definition A.1 *The* modular group, *denoted by* $\Gamma(1)$, *is the quotient group*

$$\Gamma(1) = SL_2(\mathbb{Z})/\{\pm 1\}$$

± 1 are the only elements of $SL_2(\mathbb{Z})$ which fix $\mathbf{H}$. This group is generated by the matrices

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad and \quad T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Definition A.2 *A function* $f(\tau)$ *is called a modular function of weight* $2k$ *if*

1. f *is mermorphic in upper half-plane* $\mathbf{H}$;
2. *for every* $\tau \in \mathbf{H}$ *and* $\gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma(1)$
$$f(\gamma\tau) = (c\tau + d)^{2k} f(\tau);$$
3. *the Fourier expansion*
$$f(\tau) = \sum_{i=-n}^{\infty} a_i e^{2\pi i \tau}$$
is valid throughout the upper half-plane $\mathbf{H}$.

Any modular function of weight 0 is called modular form. Hence, modular forms for all $\gamma \in \Gamma(1)$

$$f(\gamma\tau) = f(\frac{a\tau + b}{c\tau + d}) = f(\tau).$$

The modular forms are invariant under homothetic transformation of τ. For any lattice $\Lambda = \mathbb{Z}\tau + \mathbb{Z}$, the Eisenstein series

$$G_{2k}(\Lambda) = \sum_{\omega \in \Lambda, \omega \neq 0} \frac{1}{\omega^{2k}}$$

is absolutely convergent for all integers $k \geq 2$ and is a modular functions of weight $2k$.

We are mainly interested in $g_2 = 60G_4$ and $g_3 = 140G_6$. The modular discriminant is the function $\triangle(\tau) = g_2^3 - 27g_3^2$. It is a modular form of weight 12.

The Klein's modular function $j(\tau)$ is a combination of g_2 and g_3 defined in such a way that as a function of τ, it is homogeneous function of degree 0.

$$j(\tau) = \frac{g_2^3(\tau)}{\triangle(\tau)}$$

Since it is a modular form, it remains invariant under unimodular transformation of modular group $\Gamma(1)$ and is uniquely related to g_2 and g_3. The Fourier expansion of $\triangle(\tau)$ and $j(\tau)$ are as given below.

$$\begin{aligned} \triangle(\tau) &= \frac{1}{(2\pi)^{12}} \sum_{n \geq 0} \tau(n) q^n \\ j(\tau) &= \frac{1}{q} + \sum_{n \geq 0} c(n) q^n \end{aligned}$$

where, $q = e^{2\pi i \tau}$, $\tau(n)$ are Ramanujan's function and $c(n) \in \mathbb{Z}$.

Appendix B

Algebraic Number Theory

Algebraic number theory is a very vast topic in mathematics. Here our aim is to give an introductory overview of the subject. For more information please refer [PS92, Ono90, Cha88, IR82, Hec93, Coh78, Ros94].

A complex number α is called algebraic if it satisfies an equation of the form

$$\alpha^n + a_{n-1}\alpha^{n-1} + \ldots + a_n = 0$$

with $a_i \in \mathbb{Q}$.

An algebraic number field K is a finite field extension of $\mathbb{Q}$ lying in $\mathbb{C}$. It is always of the form $K = \mathbb{Q}(\alpha)$. The degree $[K : \mathbb{Q}]$ of K over $\mathbb{Q}$ is called the degree of K and it is equal to the polynomial given above. The algebraic number field K can also be thought as an n dimensional vector space over $\mathbb{Q}$. The $\mathbb{Q}$-basis $(1, \alpha, \ldots, \alpha^{n-1})$ spans the whole space, i.e. the field K.

The first goal of algebraic number theory is the extension of the arithmetic in $\mathbb{Z}$ and $\mathbb{Q}$ to algebraic number fields. By arithmetic in $\mathbb{Z}$ we mean the unique factorization of natural numbers in the product of prime numbers. The arithmetic in $\mathbb{Q}$ is given by the arithmetic in $\mathbb{Z}$. If we want to generalize the arithmetic of $\mathbb{Q}$ to an algebraic number fields K, the first question we have to answer is, what is the generalization of $\mathbb{Z}$? This is a ring $\mathcal{O}$ in K with the following properties:

1. K is the quotient field of $\mathcal{O}$.

2. $\mathcal{O} \cap \mathbb{Q} = \mathbb{Z}$.

3. The additive group of $\mathcal{O}$ is finitely generated.

A ring with these properties is called an *order* of K. If $K \neq \mathbb{Q}$, then there are infinitely many orders of K and there is one maximal order $\mathcal{O}_K$ containing all orders of K. An element α o

K belongs to $\mathcal{O}_\mathcal{K}$ if and only if there are integers $a_1, \ldots, a_s \in \mathbb{Z}$ for some $s \leq n$ such that

$$\alpha^s + a_{s-1}\alpha^{s-1} + \ldots + a_s = 0$$

The ring of integers is a natural generalization of $\mathcal{O}_\mathcal{K}$ and arithmetic in $\mathbb{Z}$ is generalized with the notion of ideals in $\mathcal{O}_\mathcal{K}$. The $\mathcal{O}_\mathcal{K}$ is called the ring of algebraic integers and its elements are called algebraic integers. The ideal theory of algebraic number field plays an important role in giving a well defined structure to arithmetic in $\mathcal{O}_\mathcal{K}$. An integral ideal in $\mathcal{O}_\mathcal{K}$ is a $\mathbb{Z}$ module in $\mathcal{O}_\mathcal{K}$. An integral ideal is also referred as a module.

Proposition B.1 *A number α in an algebraic number field K is an algebraic integer if and only if there exists a module $\mathfrak{m} \neq \{0\}$ in K with $\alpha\mathfrak{m} \subset \mathfrak{m}$.*

Let $\mathfrak{m}$ be a complete module[1] in K. Then

$$\mathcal{O}(\mathfrak{m}) = \{\alpha \in K | \alpha\mathfrak{m} \in \mathfrak{m}\}$$

is called the order of $\mathfrak{m}$.

Proposition B.2 *Any module $\mathfrak{m}$ of $\mathcal{O}$ contains a basis of the vector space K over $\mathbb{Q}$.*

For any α in K, there are n $(= [K : \mathbb{Q}])$ conjugates [Cha88, Ono90, Fra95] $\alpha^{(i)}$ $1 \leq i \leq n$. The norm and trace functions of α are defined as

$$\mathrm{N}_{K/\mathbb{Q}}(\alpha) = \alpha^{(1)} \ldots \alpha^{(n)} \quad \textit{and} \quad \mathrm{Tr}_{K/\mathbb{Q}}(\alpha) = \alpha^{(1)} + \ldots + \alpha^{(n)}$$

For $(\alpha_1, \ldots, \alpha_n)$ the discriminant is defined as

$$d_{K/\mathbb{Q}}(\alpha_1, \ldots, \alpha_n) = (det(\alpha_i^{(j)}))^2 = det(\mathrm{Tr}_{K/\mathbb{Q}}(\alpha_i\alpha_j))$$

If all $\alpha_i \in \mathcal{O}_\mathcal{K}$ then the discriminant will be an integer. A set of n numbers $(\alpha_1, \ldots, \alpha_n)$ is called an integral basis of module $\mathfrak{m}$ if we have $\mathfrak{m} = \mathbb{Z}\alpha_1 + \ldots \mathbb{Z}\alpha_n$. The discriminant of this module is independent of the choice of the basis.

Theorem B.1 *The discriminant of any order $d_{K/\mathbb{Q}}(\mathcal{O})$ can either be 0 or 1 congruent to modulo 4.*

Let $(\beta_1, \ldots, \beta_n)$ be a basis of $\mathcal{O}(\mathfrak{m})$ and let $A \in GL_n(\mathbb{Q})$ such that

$$(\alpha_1, \ldots, \alpha_n) = (\beta_1, \ldots, \beta_n)A.$$

The absolute value $|detA|$ of the determinant of A is independent of the choice of the basis. It is called the norm of $\mathfrak{m}$ and is denoted by $\mathrm{N}(\mathfrak{m})$. It is easy to see that

$$d(\mathfrak{m}) = d(\mathcal{O}(\mathfrak{m}))\mathrm{N}(\mathfrak{m})^2$$

Here $\mathrm{N}(\mathfrak{m}) = |detA|$. If $\mathfrak{m}$ is an order $\mathcal{O}$ and its order is the ring of algebraic integers then $\mathrm{N}(\mathcal{O})$ is called the conductor of $\mathcal{O}$.

[1] A module is called complete if its rank is equal to $[K : \mathbb{Q}]$

Definition B.1 *For modules (equivaletly ideal)* $\mathfrak{a}$ *and* $\mathfrak{b}$ *in an order* $\mathcal{O}$*, we define*

$$\mathfrak{a} \sim \mathfrak{b} \Leftrightarrow (\alpha)\mathfrak{a} = (\beta)\mathfrak{b} \quad \textit{for some } \alpha, \beta \in \mathcal{O},$$

where (α) & (β) are principal ideals. This relation in the set of all nonzero modules (ideals) in $\mathcal{O}$ is an equivalence relation. A class defined by this equivalence relation is called an ideal class.

Theorem B.2 *The number of ideal classes of* $\mathcal{O}$ *if finite.*

The number of ideal classes of $\mathcal{O}$ is called class number, denoted by h.

Theorem B.3 *The set of all ideal classes of* $\mathcal{O}$ *constitutes an abelian group, called class group of* $\mathcal{O}$.

Equivalence class of all principal ideals of a class group is the identity of this group. Obviously, any ideal in $\mathcal{O}$ when raised to power h will become principal ideal. In fact, this concept of class group can be extended to fractional ideals from integral ideal.

Definition B.2 *Let* K *be an algebraic number field of finite degree over* $\mathbb{Q}$ *and let* $\mathcal{O}$ *be an order. An* $\mathcal{O}$*-module* $\mathfrak{m}$ *is a fractional ideal if there is a* $\mu \in \mathcal{O}$ *such that* $\mu\mathfrak{m} \subset \mathcal{O}$.

If the set of all principal fractional ideals is denoted by P and class group by H, then

Theorem B.4 *The group* H/P *is same as the class group defined above.*

The next theorem characterizes the arithmetic in K.

Theorem B.5 *(Fundamental Theorem of Ideal Theory) Every nontrivial fractional ideal* $\mathfrak{a}$ *can be written uniquely as a product of prime ideals except for the order of the factors:*

$$\mathfrak{a} = \mathfrak{p}_1^{e_1}, \ldots, \mathfrak{p}_g^{e_g} \quad e_i \in \mathbb{Z}$$

Hence for any rational prime p the principal ideal $(p)_K$ in $\mathcal{O}$ splits into prime ideals

$$(p)_K = \mathfrak{p}_1^{e_1} \ldots \mathfrak{p}_g^{e_g}; \quad e_i > 0$$

Call f_i the degree of $\mathfrak{p}_i$: $f_i \stackrel{def}{=} [\mathcal{O}/\mathfrak{p}_i : GF(p)]$. Here

$$\mathrm{N}\mathfrak{p}_i = [\mathcal{O}/\mathfrak{p}_i] = p^{f_i}.$$

Taking the norm of both the sides and comparing the power of p, we get

$$n = [K : \mathbb{Q}] = e_1 f_1 + \cdots + e_g f_g.$$

Definition B.3 *e_i is the ramification index of $\mathfrak{p}_i$. When $e_i > 1$, $\mathfrak{p}_i$ is said to be ramified and when $e_i = 1$, $\mathfrak{p}_i$ is unramified.*

This concept can be generalized further to the extension field L of K. Now we focus upon the ring of algebraic integers but all the arguments will be valid for orders in the extension fields as well. For a prime p in $\mathbb{Q}$, denote by $\mathfrak{p}$ and $\mathfrak{P}$ prime ideals in $\mathcal{O}_\mathcal{K}$ and $\mathcal{O}_\mathcal{L}$, respectively, such that $\mathfrak{p}|p$ and $\mathfrak{P}|\mathfrak{p}$. Call e the ramification index of $\mathfrak{p}$ for $K/\mathbb{Q}$, e^* the ramification index of $\mathfrak{P}$ for $L/\mathbb{Q}$. Furtther more, let e' be the ramification index of $\mathfrak{P}$ for L/K. Hence

$$\begin{cases} (p)_K = \ldots \mathfrak{p}^e \ldots \\ (p)_L = \ldots \mathfrak{P}^{e^*} \ldots \\ (\mathfrak{p})_K = \ldots \mathfrak{P}^{e'} \ldots \end{cases}$$

As for their degrees, we have
$f = [\mathcal{O}_\mathcal{K}{:}GF(p)]$ and $f^* = [\mathcal{O}_\mathcal{L}{:}GF(p)] = [\mathcal{O}_\mathcal{L}/\mathfrak{P}{:}\ \mathcal{O}_\mathcal{K}/\mathfrak{p}][\mathcal{O}_\mathcal{K}/\mathfrak{p}{:}GF(p)] = f'f$,
where $f' = [\mathcal{O}_\mathcal{L}/\mathfrak{P}{:}\mathcal{O}_\mathcal{K}/\mathfrak{p}]$. Therefore we obtain $f^* = f'f$.

Now we give a very impotant result which tells us whether any prime will ramify in the extension field or not.

Theorem B.6 *Any prime ideal $\mathfrak{p}$ in K will ramify in L if discriminant of L $(= d(\mathcal{O}_\mathcal{L}))$ is divisible by $\mathfrak{p}$.*

The next theorem, is due to Kummer, is very important in determining the irreducibility of the class equation constructed in Chapter 4.

Theorem B.7 *Let $K = \mathbb{Q}(\theta)$, $\theta \in \mathcal{O}_\mathcal{K}$ and let $f_\theta(x) \in \mathbb{Z}[x]$ be the minimal polynomial of θ. Assume that $\mathcal{O}_\mathcal{K} = \mathbb{Z}[\theta]$. Let*

$$f_\theta(x) \equiv \varphi_1(x)^{e_1}, \ldots, \varphi_g(x)^{e_g} \ (mod\ p)$$

be the decomposition of $f_\theta(x)$ mod p where each $\varphi_i(x)$ is a monic polynomial of degree f_i and irreducible mod p. Then $\mathfrak{p} = (p, \varphi_i(\theta))$ is a prime ideal of degree f_i and we have

$$(p)_K = \mathfrak{p}_1^{e_1}, \ldots, \mathfrak{p}_g^{e_g}$$

Since our major concern is with imaginary quadratic fields, we summarize the above discussion for degree two extensions. An algebraic field K is called a quadratic field if $[K : \mathbb{Q}] = 2$. For any quadratic field there exists a unique square free $d \in \mathbb{Z}$ such that $K = \mathbb{Q}(\sqrt{d})$. Let

$$\omega = \begin{cases} (1 + \sqrt{d})/2 & if\ d \equiv 1\ mod\ 4, \\ \sqrt{d} & if\ d \equiv 2, 3\ mod\ 4 \end{cases}$$

Then $\{1,\omega\}$ is a basis of $\mathcal{O}_K$.
An arbitrary order in K is of the form $\mathcal{O}_f := \mathbb{Z}[f\omega]$, where f is a positive integer f, called the conductor of $\mathcal{O}_f$. The basis of $\mathcal{O}_f$ is $\{1,f\omega\}$ The discriminant of $\mathcal{O}_K$ is called the discriminant of K and is equal to $D = d$ if $d \equiv 1 \ mod\ 4$ and $D = 4d$ if $d \equiv 2,3 \ mod\ 4$. If the discriminant is negative then the field is called an imaginary quadratic field. The discriminant of an order with conductor f, $\mathcal{O}_f$, is equal to Df^2. The class number of $\mathcal{O}_f$ is an integer multiple of that of $\mathcal{O}_K$.
If $d \equiv 2,3 \ mod\ 4$ and p is a prime, then

1. $p|D \Rightarrow (p) = \mathfrak{p}^2$, and $N(\mathfrak{p}) = p$.

2. for $p \not| D$ if D is a quadratic residue modulo p, then $(p) = \mathfrak{p}\mathfrak{p}'$ and $N(\mathfrak{p}) = N(\mathfrak{p}') = p$. If D is not a quadratic residue modulo p, then $(p) = \mathfrak{p}$ and $N(\mathfrak{p}) = p^2$.

If $d \equiv 1 \ mod\ 4$, then

1. for $p|D \Rightarrow (p) = \mathfrak{p}^2$, and $N(\mathfrak{p}) = p$.

2. for $p \not| D$,

   (a) $p \neq 2$: If D is quadratic residue modulo p, then $(p) = \mathfrak{p}\mathfrak{p}'$ and $N(\mathfrak{p}) = N(\mathfrak{p}') = p$. If D is not a quadratic residue modulo p, then $(p) = \mathfrak{p}$ and $N(\mathfrak{p}) = p^2$.

   (b) $p = 2$. If $D \equiv 1 \ mod\ 8$, then $(2) = \mathfrak{p}\mathfrak{p}'$ and $N(\mathfrak{p}) = N(\mathfrak{p}') = p$. If $D \equiv 5 \ mod\ 8$, then $(2) = \mathfrak{p}$ and $N(\mathfrak{p}) = 2^2$.

Appendix C

Class Field Theory

Theory of abelian extensions of algebraic number fields is called the class field theory. It describes the arithmetic of abelian extension of an algebraic number field K in terms of arithmetic of K. In 1840 it was shown by Kummer that every abelian extension of $\mathbb{Q}$ is contained in a cyclotomic field $\mathbb{Q}(\zeta_m)$, where ζ_m is the mth root of unity. The theory of complex multiplication provides an analytical realization of class field theory for quadratic imaginary fields, much as the cyclotomic theory gives a realization of class field theory for $\mathbb{Q}$. Here we give a brief overview of the class field theory. Please refer [Coh78, Sil94, PS92, Ono90] for further details.

Class field theory describes the arithmetic properties of an abelian extension of K with the corresponding Galois group of automorphisms. For any abelian extension of L of K, the Galois group is denoted by $Gal(L/K)$ and $\#Gal(L/K) = [L : K]$. For a prime ideal $\mathfrak{p}$ in K consider the finite set of prime ideals in $\mathcal{O}_{\mathcal{L}}$;

$$P_{\mathfrak{p}} = \{\mathfrak{P}:\mathfrak{P}|\mathfrak{p}\}$$

The decomposition group of $Gal(L/K)$ for $\mathfrak{P}|\mathfrak{p}$ is the set of all automorphisms which leaves $\mathfrak{P}$ fixed, i.e.

$$G(\mathfrak{P}) \stackrel{def}{=} \{\sigma \in Gal(L/K) : \mathfrak{P}^{\sigma} = \mathfrak{P}\}.$$

The quotient group $Gal(L/K)/G(\mathfrak{P})$ is isomorphic to $P_{\mathfrak{P}}$. Let $g = [P_{\mathfrak{P}}] = [Gal(L/K) : G(\mathfrak{P})]$. Let $\sigma_1, \ldots, \sigma_g$ be the total set of representatives of the left cosets in $Gal(L/K)/G(\mathfrak{P})$. Then the decomposition of $\mathfrak{p}$ in L can be written as

$$\mathfrak{p} = (\mathfrak{P}^{\sigma_1})^{e_1} \ldots (\mathfrak{P}^{\sigma_g})^{e_g}$$

Now since for any $\sigma \in Gal(L/K)$, $\mathfrak{p}^{\sigma} = \mathfrak{p}$ and $Gal(L/K)$ is abelian, it can easily be shown that all e_i will be equal. Hence

$$\mathfrak{p} = (\mathfrak{P}^{\sigma_1}, \ldots, \mathfrak{P}^{\sigma_g})^{e}$$

As for the residue fields, there is a natural bijection

$$\mathcal{O}_{\mathcal{L}}/\mathfrak{P} \approx \mathcal{O}_{\mathcal{L}}/\mathfrak{P}^{\sigma} \quad \sigma \in Gal(L/K)$$

and so the relative degree $[\mathcal{O}_{\mathcal{L}}/\mathfrak{P}^{\sigma} : \mathcal{O}_{\mathcal{K}}/\mathfrak{p}]$ does not depend upon σ. If the relative degree is f then

$$n = [L : K] = efg$$

Now there exists a homomorphism from the decomposition group of $\mathfrak{P}$ to the Galois group of residue fields $\mathcal{O}_{\mathcal{L}}/\mathfrak{P}$ and $\mathcal{O}_{\mathcal{K}}/\mathfrak{p}$,

$$G(\mathfrak{P}) \longrightarrow Gal((\mathcal{O}_{\mathcal{L}}/\mathfrak{P})/(\mathcal{O}_{\mathcal{K}}/\mathfrak{p}))$$

The right hand side group is cyclic and generated by the Frobenius automorphism

$$x \longmapsto x^{\mathrm{N}_{\mathbb{Q}}^{K}(\mathfrak{p})}.$$

If $\mathfrak{p}$ is unramified, i.e. $e = 1$, then this homomorphism will be an isomorphism and there will exist a unique element $\sigma_{\mathfrak{p}} \in G(\mathfrak{P})$ (and hence in $Gal(L/K)$) which will correspond to Frobenius automorphism in the right hand side group. It should be noted that this will be a unique element in case of abelian extensions only. Hence $\sigma_{\mathfrak{p}} \in Gal(L/K)$ is uniquely determined by the condition

$$\sigma_{\mathfrak{p}}(x) = x^{\mathrm{N}_{\mathbb{Q}}^{K}(\mathfrak{p})} \ mod\ \mathfrak{P} \quad \text{for all } x \in L$$

This completes the study of Galois group of abelian extension of an algebraic number field K. Now we discuss relationship of this group with the ideal class group of ring of integer $\mathcal{O}_{\mathcal{K}}$ of K (and also for any order $\mathcal{O}$ in general).

Let $\mathfrak{c}$ be an integral ideal of K that is divisible by all primes that ramify in L/K, and let

$$I(\mathfrak{c}) = \text{group of fractional ideals of } K \text{ which are relatively prime to } \mathfrak{c}.$$

Then the *Artin Map* is defined using the $\sigma_{\mathfrak{p}}$'s

$$(\cdot,\, L/K) : I(\mathfrak{c}) \longrightarrow Gal(L/K),$$
$$(\mathfrak{a},\, L/K) = (\prod_{\mathfrak{p}} \mathfrak{p}^{n_{\mathfrak{p}}},\, L/K) \stackrel{def}{=} \prod_{\mathfrak{p}} \sigma_{\mathfrak{p}}^{n_{\mathfrak{p}}}.$$

The following proposition gives us important information about Artin map.

Proposition C.1 *(Artin Reciprocity) Let L be a finite abelian extension of K. There exis an integral ideal $\mathfrak{c} \subset \mathcal{O}_{\mathcal{K}}$, divisible by precisely the primes of K that* ramify *in L, such that*

$$((\alpha),\, L/K) = 1 \quad \textit{for all } \alpha \in K^{*} \textit{ satisfying } \alpha \equiv 1 (mod\ \mathfrak{c}).$$

$\mathfrak{c}$ is called *conductor* of L/K. In view of this proposition, we define the group of principal ideals which are relatively prime to $\mathfrak{c}$:

$$P(\mathfrak{c}) = \{(\alpha) : \alpha \in K^*,\ \alpha \equiv 1 (\mathrm{mod}\ \mathfrak{c})\}.$$

Artin reciprocity says that the kernel of the Artin map contains $P(\mathfrak{c})$ for an appropirate choice of $\mathfrak{c}$. More precisely,

$$\mathfrak{a} \in P(\mathfrak{c}) \Longrightarrow (\mathfrak{a},\ L/K) = 1.$$

If $\mathfrak{p}$ is a prime ideal in K such that it is unramified in L, then $\mathfrak{p}$ splits completely in L if and only if the degree of corresponding ideals in L is 1, or equivalently $(\mathfrak{P},\ L/K) = 1$. Thus the unramified prime ideals in the kernel of the Artin map are precisely the primes of K which split completely in L. Now, we introduce the notion of class field with *Ray Class Field.*

Definition C.1 *Let $\mathfrak{c}$ be an integral ideal of K. A ray class field of K (modulo $\mathfrak{c}$) is a finite abelian extension $K_{\mathfrak{c}}/K$ with the property that if the conductor of any finite abelian extension L/K devide $\mathfrak{c}$ then $K \subset K_{\mathfrak{c}}$.*

The ray class fields are the largest fields with a given conductor. Consider the ray class field of K modulo the unit ideal $\mathfrak{c}$= (1) (= $\mathcal{O}_K$). Then all the prime ideals of K, if at all, will be unramifield in $L = K_{(1)}$. In such a case, L is called the *Hilbert Class field*, the maximal unramified abelian extension K. Notice that

$$\begin{aligned} I((1)) &= \{\text{all non-zero fractional ideals of } K\}, \\ P((1)) &= \{\text{all non-zero fractional ideals of } K\}. \end{aligned}$$

So the Artin map induces an isomorphism between the ideal class group of K and the Galois group of the Hilbert class field:

$$(\cdot,\ K_{(1)}/K) : \mathcal{CL}(\mathcal{O}_K) \cong Gal(K_{(1)}/K).$$

Similarly for any conductor (f), where $f \in \mathbb{Z}$, all of the above arguments hold for an order $\mathcal{O}$ with conductor f (see Appendix B) and

$$(\cdot,\ K_{(f)}/K) : \mathcal{CL}(\mathcal{O}) \cong Gal(K_{(f)}/K).$$

In this case, $K_{(f)}$ (denoted as $\mathcal{H}_{\mathcal{O}}$ in Chapter 2 and 4) is called the ring class field. This is not an unramified abelian extension of K as all prime ideals of $\mathcal{O}_K$ which are not relatively prime to (f) will surely ramify in $K_{(f)}$. But (obviously) all the prime ideals in $\mathcal{O}$ will be unramified in $K_{(f)}$. This result is of utmost importance in the theory of construction of elliptic curves.

Appendix D

Algebraic Geometry

Let K be a perfect field [Fra95, BJN94] and $\overline{K}$ its algebraic closure.

Definition D.1 *An affine n-space over K is the set of n-tuples*

$$\mathbb{A}^n(\overline{K}) = \{\mathcal{P} = (x_1, x_2, \ldots, x_n) : x_i \in \overline{K}\}$$

Similarly, the set of K-rational point in $\mathbb{A}^n$ *is the set*

$$\mathbb{A}^n(K) = \{\mathcal{P} = (x_1, x_2, \ldots, x_n) \in \mathbb{A}^n(\overline{K}) : x_i \in K\}$$

The elements of $\mathbb{A}^n(\overline{K})$ are called points. $\mathbb{A}^1(\overline{K})$ is the affine line and $\mathbb{A}^n(\overline{K})$ is an affin plane. For any function f in the ring $\overline{K}[X_1, \ldots, X_n]$, a point $\mathcal{P}= (a_1, a_2, \ldots, a_n) \in \mathbb{A}^n(\overline{K}$ is a zero of f if $f(\mathcal{P}) = 0$. If f is not a constant, then the set of all zeros of f is called **hypersurface** defined by f, and is denoted by $V(f)$. A hypersurface in $\mathbb{A}^2(\overline{K})$ is called a affine plane curve. More, generally if S is any set of polynomials in $\overline{K}[X_1, \ldots, X_n]$, we le $V(S) = \{\mathcal{P} \in \mathbb{A}^n(\overline{K}) : f(\mathcal{P}) = 0 \ \ for\ all f \in S\}$.

Definition D.2 *A subset* $X \subset \mathbb{A}^2(\overline{K})$ *is an affine algebraic set, or simply an algebraic se*

We can associate an ideal with this algebraic set, called ideal of V, which is given as $I(V) =$ $\{f \in \overline{K}[X_1, \ldots, X_n] : f(\mathcal{P}) = 0 \ for\ all\ \mathcal{P} \in V\}$. Any algebraic set $V(f)$ is said to b irreducible if $I(V(f))$ is a prime ideal in $\overline{K}[X_1, \ldots, X_n]$.

Definition D.3 *An irreducible affine algebraic set is called an affine variety.*

The affine coordinate ring and function field of an affine vareity $V(f)$ over K are give by

$$K[V] = \frac{K[X_1, \ldots, X_n]}{I(V(f))} \ \ and \ \ K(V) = \{a/b : b \neq 0, a, b \in K[V]\}$$

Now, we define the projective space as the collection of lines in affine space of one high dimension. Each point in an affine plane is represented by a line passing through orig

in the corresponding projective plane (affine plane of one higher dimension). Actually, the projective plane arose through the process of adding "point at infinity" to the affine space.

Definition D.4 *A projective n-space over K is the set of $n+1$-tuples*

$$\mathbb{A}^{n+1}(\overline{K}) = \{\mathcal{P} = (x_1, x_2, \ldots, x_{n+1}) : x_i \in \overline{K}\}$$

such that at least one tuple is non-zero. Similarly, the set of K-rational point in $\mathbb{A}^n$ is the set

$$\mathbb{A}^n(K) = \{\mathcal{P} = (x_1, x_2, \ldots, x_{n+1}) \in \mathbb{A}^n(\overline{K}) : x_i \in K\}$$

Two points in a projective plane are said to be equivalent if their coordinates are $\{x_i\}$ and $\{\lambda x_i\}$ for any $\lambda \in \overline{K}$. In other words, all the poitns lying on a line which passes through origin in projective plane will be equivalent. In fact, all of these points will be equivalent to a single point in the affine plane given by $\{x_1/x_i, \ldots, x_{i-1}/x_i, x_{i+1}/x_i, \ldots, x_{n+1}/x_i\}$.

Prjective variety is defined in a way similar to the affine vareity. Any affine vareity (hypersurface) $V(f)$ in an affine plane corresponds to a projective variety $V'(f')$ in a projective plane for homogenized version f' of f. The next theorem tells us about the isomorphism between the two varieties.

Theorem D.1 *Two projective (also affine) vareities V_1, V_2 are said to be isomorphic (or birationally equivalent) iff there exist a map (set of rational functions in corresponding function field) which maps one variety to another.*

$$\phi = [f_1, \ldots, f_{n+1}] : V_1 \longmapsto V_2 \quad \textit{where } f_i \in \overline{K}(V)$$

Since we are mainly concerned with algebraic curves (an affine variety of dimension 1) , now onward, we will concentarte on affine plane $\mathbb{A}^2(K)$ and corresponding projective plane $\mathbb{A}^3(K)$ only. Any curve $C : f(x, y) = 0$ will correspond to a hypersurface $C' : f'(X, Y, Z) = f(X/Z, Y/Z) = 0$.

Now, let us define the term non-singular (or smooth) for algebraic curves (or any variety of higher dimension) which is a measure of irregularity of the curve (of the variety).

Definition D.5 *A projective curve $C = V(f)$ is said to be non-singular or smooth if all $\frac{\partial f}{\partial X_i}$ are not zero simultaneously for any point of C.*

If a curve is singular, then it is possible to draw more than one tanegent at singular points. The number of distinct tangents which can be drawn on an ordinary singlular point, define its multiplicity. It is well known in algebraic geometry that any curve of degree n (the degree of corresponding f) can have at the most (n-1)(n-2)/2 double points (ordinary singular points of multiplicity 2). The *genus* of a curve $g = (n-1)(n-2)/2 -$

(*number of properly counted double points*) is a measure of singularity of a curve. For more information about genus please see [Ful69, Abh90, Wal62, Mor93, PS93, TV91].

To conclude this discussion, we define the *divisor group* for a smooth curve. The divisor group of a curve C, denoted by $Div(C)$, is the free abelian group generated by the formal sum of points of C. Thus, a divisor $D \in Div(C)$ is a formal sum

$$D = \sum_{\mathcal{P} \in C} n_{\mathcal{P}}(\mathcal{P})$$

with $n_{\mathcal{P}} \in \mathbb{Z}$ and $n_{\mathcal{P}} \neq 0$ for all but finitely many $\mathcal{P} \in C$. The degree of D is defined by

$$deg\ D = \sum_{\mathcal{P} \in C} n_{\mathcal{P}}.$$

Divisors of degree 0 form a subgroup of $Div(C)$, which is denoted by

$$Div^0(C) = \{D \in Div(C) : deg = 0\}$$

If curve is defined over K, then corresponding divisor groups will be $Div_K(C)$ and $Div_K^0(C)$. Now, assume that the curve C is smooth, and let f lie in the function field of C, then we can associate to f the divisor $\mathrm{div}(f)$ given by

$$\mathrm{div}(f) = \sum_{\mathcal{P} \in C} \mathrm{ord}_{\mathcal{P}}(f)(\mathcal{P}),$$

where, $\mathrm{ord}_{\mathcal{P}}(f)$ is a valuation on f for its poles and zeros. If f has a pole at $\mathcal{P}$, $\mathrm{ord}_{\mathcal{P}}(f) < 0$; if $\mathcal{P}$ is zero of f then $\mathrm{ord}_{\mathcal{P}}(f) > 0$. Since f is an element of function field of C, there will be finitely many points at which f will have a pole or zero. Moreover, it is a fundamental fact in algebraic geometry that degree of $\mathrm{div}(f)$ will be zero.

Definition D.6 *A divisor $D \in Div(C)$ is principal if it has the form $D = div(f)$ for some f in function field $\overline{K}(C)^*$. Two divisors D_1, D_2 are linearly equivalent, denoted by $D_1 \sim D_2$, if $D_1 - D_2$ is principal. The divisor class group (or Picard group) of C, denoted $Pic(C)$, is the quotient of $Div(C)$ by the subgroup of principal divisors. We let $Pic_K(C)$ be the subgroup of $Pic(C)$ fixed by Galois group $Gal(\overline{K}/K)$. The degree 0 part of the divisor class group of C, $Pic^0(C)$, is the quotient of $Div^0(C)$ by the subgroup of principal divisors. Further, $Pic_K^0(C)$ is the subgroup of $Pic^0(C)$ fixed by $Gal(\overline{K}/K)$*

This definition is helpful in defining the group structure of all the point on a given elliptic curve.

Bibliography

[Abh90] Shreeram S. Abhyankar. *Algebraic Geometry for Scientists and Engineers.* Number 35. American Mathematical Society, 1990.

[ABV89] David W. Ash, I. F. Blake, and S.A. Vanstone. Low Complexity Normal Bases. *Discrete Applied Mathematics*, 25:149–210, 1989.

[Ahl79] Lars V. Ahlfors. *Complex Analysis.* McGraw-Hill, 1979.

[AM93] A. Atkin and F. Morain. Elliptic Curves and Primality Proving. *Mathematics of Computation*, 61(203):29–68, 1993.

[AMOV91] G. Agnew, R. Mullin, I. Onyszchuk, and S. Vanstone. An Implementation for a Fast Public Key cryptosystem. *Journal of Cryptography*, (3):63–79, 1991.

[AMV93] G. Agnew, R. Mullin, and S. Vanstone. An Implementation of elliptic Curve Cryptosystems over F_{2^m}. *IEEE Journal on Selected Areas in Communications*, 11(5):804–813, June 1993.

[Apo76] Tom M. Apostol. *Modular Functions and Dirichlet Series in Number Theory.* Number GTM-41. Springer-Verlag, 1976.

[BCh+66] A. Borel, S. Chowla, C.S. herz, K. Iwasawa, and J-P. Serre. *Seminar on Complex Multiplication.* Number LNM-21. Springer-Verlag, 1966.

[BJN94] P.B. Bhattacharya, S.K. Jain, and S.R. Nagpaul. *Basic Abstract Algebra.* Cambridge University Press, 1994.

[Bri89] E. F. Brickell. A Survey of Hardware Implementation of RSA. *LNCS: Advances in Cryptology-CRYPTO'89*, (435):368–370, 1989.

[BS96] Erich Bach and Jeffrey Shallit. *Algorithmic Number Theory*, volume 1: Efficient Algorithms. The MIT Press, 1996.

[Cas94] *32-bit Microcontroller for Smart Cards*, 1994. available at site http://www.dice.ucl.ac.be/dhem/cascade.html.

[Cha88] J. S. Chahal. *Topics in Number Theory.* Plenum Press, 1988.

[Cha95] K.P.P.Kalyan Chakravarthy. On Certain Computations Related to Elliptic Curves. Master's thesis, Indian Institute of technology, kanpur, Electrical Engg. Deptt., IIT,. Kanpur-208016,India, February 1995.

[Coh78] Harvey Cohn. *A Classical Invitation to Algebraic Numers and Class Fields.* Springer-Verlag, 1978.

[CTT94] Jinhui Chao, Kazuo Tanada, and Shigeo Tsujii. Design of elliptic Curves with Controllable Lower Boundary of Extension Degree For Reduction Attacks. *LNCS:Advances in Cryptology-CRYPTO'94*, 839:50–55, 1994.

[Dav80] Harold Davenport. *Multiplicative Number Theory.* Springer-Verlag, 1980.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptgraphy. *IEEE Transaction on Information Theory*, 22(6):644–654, November 1976.

[DH79] Whitfield Diffie and Martin E. Hellman. Privacy and Authentication: An Introduction to Cryptography. *Proceeding of The IEEE*, 67(3):397–427, March 1979.

[DJ91] Stephen R. Dusse and Burton S. Kaliski Jr. A Cryptographic Library for the Motorola DSP56000. *LNCS:Advances in Cryptology-EUROCRYPT'90*, 473:230–244, 1991.

[DVJ96] Jean-Francois Dhem, Daniel Veithen, and J.J.Quisaquater. SCALP:Smart Cards For Limited Payment Systems. *IEEE Micro*, pages 42–51, June 1996.

[dWQ92] Domnique de Waleffe and J.J. Quisquater. CORSAIR: A Smart Card for Public Key Cryptosystems. *LNCS:Advances in Cryptology'92*, pages 389–399, 1992.

[ElG85] T. ElGamal. A public key Cryptsystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transaction on Information Theory*, 31:469–472, 1985.

[Fen89] Gui-Liang Feng. A VLSI Architecture for Fast Inversion in $GF(2^n)$. *IEEE Transaction on Computers*, 38(10):1383–1386, October 1989.

[FOM92] Atsushi Fujioka, Tatsuaki Okamato, and Shoji Miyaguchi. ESIGN: An Efficient Digital Signature Implementation for Smart Cards. *LNCS:Advances in Cryptology'92*, pages 389–399, 1992.

[Fra95] John B. Fraleigh. *A First Course in Abstract Algebra.* Narosa Publishing House, 1995.

[Ful69] William Fulton. *Algebraic Curves.* W. A. Benjamin, Inc, 1969.

[Gre93] Jonathan S. Greenfield. Distributed Programming Paradigms in cryptography applications. *Lecture Notes in Computer Science*, 870, 1993.

[GV95] Shuong Gao and S.A. Vanstone. On Orders of Optimal Normal Basis Generator. *Mathematics of Computation*, 64(211):1227–1233, July 1995.

[Hec93] Erich Hecke. *Lectures in Theory of Algebraic Numbers.* Number GTM-77. Springer-Verlag, 1993.

[IR82] Kenneth Ireland and Michael Rosen. *A Classical Introduction to Modern Number Theory.* Number GTM-84. Springer-Verlag, 1982.

[KABSK96] Cetin Kaya Koc, Tolga Acar, and Jr. Burton S. Kaliski. Analyzing and Comparing Mongomery Multiplication Algorithms. *IEEE Micro*, pages 26–33, June 1996.

[Knu81] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms.* Number MA. Addison-Wesley, Reading, 1981.

[Kob90] N. Koblitz. Constructing Elliptic Curves Cryptosystems in Characteristic 2. *LNCS: Advances in Cryptology-CRYPTO'90*, 537:156–166, 1990.

[Kon91] Hans-Peter Konigs. Cryptgraphic Identification Methods for Smart Cards in the Process of Standardization. *IEEE Communication Magazine*, pages 42–48, June 1991.

[KT92] Kenji Koyama and Yukio Tsuruoka. Speeding up Elliptic Cryptosystems by Using a Signed Binary Window Method. *LNCS:Advances in Cryptology-CRYPTO'92*, pages 345–357, 1992.

[LN94] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications.* Cambridge Unversity Press, 1994.

[LZ94] G.J. Lay and H.G. Zimmer. Constructing Elliptic Curves with Given Group Order over Large Finite Fields. *LNCS: Algorithmic Algebraic Number Theory*, (877):251–263, 1994.

[McE87] R. J. McElice. *Finite Fields for computer scientists and Engineers.* Kluwer Academic Publications, 1987.

[Men93a] Alfred J. Menezes, editor. *Application of Finite Fields.* Kluwer Academic Publishers, 1993.

[Men93b] Alfred J. Menezes. *Elliptic Curve Public Key Cryptsystems.* Kluwer Academic Publishers, 1993.

[Miy91] Atsuko Miyaji. On Ordinary Elliptic Curve Cryptosystems. *LNCS:Advances in cryptology-ASIACRYPT'91*, 1991.

[Miy92] Atsuko Miyaji. Eliptic Curve over F_p Suitable for Cryptosystems. *LNCS:Advances in Cryptology:AUSCRYPT'92*, 950:389–399, 1992.

[Mon85] P.L. Montegomery. Modular Multiplicatio Without Trial Division. *Mathematics of Computation*, 44(170):519–521, 1985.

[Mor93] Carlos J. Moreno. *Algebraic Curves over Finite Fields.* Number Cambrdge tracts in Mathematics-97. Cambridge University Press, 1993.

[MOVW89] R.C. Mullin, I.M. Onyszchuk. S.A. Vanstone, and R.M. Wilson. Optimal Normal Bases in $GF(p^n)$. *Discrete Applied Mathematics*, 22:149–161, 1989.

[MS91] Willi Meier and Othmar Staffelbach. Efficient Multiplication on Certain Nonsupersingular Elliptic Curves. *Lecture Notes in Computer Science*, pages 333–344, 1991.

[NM95] David Naccache and David M'Raihi. Cryptographic Smart Cards. *IEEE Micro*, pages 14–24, June 1995.

[Omu90] Jim K. Omura. Novel Applications of Cryptgraphy in Digtal Communications. *IEEE Communication Magazine*, pages 21–28, May 1990.

[Ono90] Takashi Ono. *A Introduction to Algebraic Number Theory.* Plenum Press, 1990.

[OSA92] Holger Orup, Erik Svendsen, and Erik Andreasen. VICTOR: An Efficient RSA Hardware Implementation. *LNCS:Advances in Cryptology'92*, pages 245–252, 1992.

[PH78] S. Pohlig and M. Hellman. An Improved Algorithm for Computing Logarithms over $GF(p)$ and its Cryptographic Significance. *IEEE Transaction on Information theory*, 24:106–110, March 1978.

[Pol78] J. Pollard. Monte Carlo Methods for Index Computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

[PS92] A. N. Parshin and I.R. Shafervich. *Number Theory II: Algebraic Number Theory.* Number EMS-60. Springer-Verlag, 1992.

[PS93] A. N. Parshin and I.R. Shafervich. *Number Theory I: Fundamental Problems, Ideas and Theories.* Number EMS-49. Springer-Verlag, 1993.

[Roo95] Peter De Rooij. Efficient Exponentiation Using Precomputations and Vector Addition Chains. *LNCS:Advances in Cryptology-EUROCRYPT-94*, 950:389–399, 1995.

[Ros94] H. E. Rose. *A Course in Number Theory.* Clarendon press, 1994.

[Sch85] Rene Schoof. Elliptic Curves over Finite Fields and the Computation of Square Roots mod p. *Mathematics of Computation*, 44(170):483–494, April 1985.

[Sch89] C.P. Schnorr. Efficient Identification and Signatures for Smart Cards. *LNCS:Advances in Cryptology-CRYPTO'89*, pages 4222–4228, 1989.

[Sch93] Bruce Schnneier. *Applied Cryptography.* John Wiley & Sons, Inc., 1993.

[Seg80] Sanford L. Segal. *Nine Introduction in Complex Analysis.* Number North-Holland Mathematics Studies-53. North-Holland Publishing Company, 1980.

[Shi71] G. Shimura. *Introduction to the Arithmetic Theory of Automorphic Forms.* Princeton University Press, 1971.

[SHL84] C. P. Scnoorr and Jr. H.W. Lenstra. A Monte Carlo Factoring Algorithm with Linear Storage. *Mathematics of Computation*, 43(167):289–31, July 1984.

[Sil85] Joseph H. Silverman. *The Arithmatic of Elliptic Curves.* Number GTM-106. Springer-Verlag, 1985.

[Sil94] Joseph H. Silverman. *Advanced Topics in the Arithmatic of Elliptic Curves.* Number GTM-151. Springer-Verlag, 1994.

[Sim91] G. Simmons, editor. *Contemporary Cryptography: The Science of Information Integrity.* IEEE Press,New York, 1991.

[Sta95] William Stalling. *Network and Internetwork Security Principles and Practice.* IEEE Press, Prentice Hall, Englewood Cliffs, New Jersy 07632, 1995.

[Ste85] N.M. Stephens. Lenstra's Factorization Method Based on Elliptic Curves. *LNCS:Advances in Cryptology-CRYPTO'85*, 218:409–416, 1985.

[TV91] M. A. Tsfaman and S.G Vlădut. *Algebraic Geometric Codes.* Number Mathematics and its Applications. Soviet Series-58. Kluwer Academic Publishers, 1991.

[VVDJ92] Andre Vandemeulebroecke, Etienne Vanzieleghem, Tony Denayer, and Paul G. A. Jespers. A Single Chip 1024 Bits RSA Processor. *LNCS: Advances in Cryptology'92*, pages 219–236, 1992.

[Wal62] R.J. Walker. *Algebraic Curves.* Dover, New York, 1962.

[Web02] H. Weber. *Lehrbuch der Algebra*, volume I,II,III. Chelsea, New York, 1902.

[WP90] Charles C. Wang and Dingyi Pei. A VLSI Design for Computing Exponentiation in $GF(2^n)$ and Its Application to Generate Pseudorandom Numbers Sequences. *IEEE Trnsaction on Computers*, 39(2):7258–262, February 1990.

[WTS+85] Charles C. Wang, T.K Truong, Howard M. Shao, Leslie J. Deutsch, Jim K. Omura, and Irving K. Reed. VLSI Architecture for Computing Multiplications and Inverses in $GF(2^n)$. *IEEE Trnsaction on Computers*, C-34(8):709–716, August 1985.

[Wun83] M.C. Wunderlich. A Performance Analysis of a Simple Prime-Testing Algorithm. *Mathematics of Computation*, 40(162):709–714, March 1983.

[Zur94] Dan Zuras. More on Squaring and Multiplying Large Integers. *IEEE Transactions on Computers*, 43(8):899–908, August 1994.

EE-1997-M-BAN-CON